Solving Quaternion Quadratic Equations (working paper 01)

Peter Michael Jack*
Independent Researcher.
Toronto, Canada †
(Dated: October 21, 2019)

We demonstrate a method for solving quadratic equations in quaternions, using the *two-hand quaternion algebra* and *rescaling parameters* in a special *unit magnitude* solution obtained by first converting the quadratic equation in one variable into a pair of linear equations in two variables.

1. QUADRATIC EQUATIONS.

#1. A first quadratic equation.

Consider the following quadratic equation $(a, b, c, q \in \mathbb{H}_R)$[1],

$$aq^2 + bq + c = 0 \qquad , \quad b \neq 0 \tag{1.1}$$

where, a, b, and c, are known quaternion parameters, and, q, is the unknown quaternion we seek to find. First, consider the special situation where the values of the known parameters are such that there exists a solution to this equation with unit magnitude, i.e., $|q| = 1$. In this case, for this particular solution, q, the inverse quaternion is just its conjugate, $1/q = q^*/|q|^2 = q^*/1 = q^*$, and so by dividing the equation throughout on the right by, q, we obtain a linear equation in the two variables, q and q^*. Then, conjugating this linear equation, yields a second linear equation, allowing us to set up a system of linear equations in two unknowns to replace the single quadratic equation in one unknown. By this procedure, we obtain the following pair of simultaneous linear equations,

$$aq + cq^* = -b \tag{1.2}$$
$$qc^* + q^*a^* = -b^* \tag{1.3}$$

In previous papers ([PJ2][PJ3][PJ4]), we discussed how to solve such systems of linear equations using the method of *two-hand quaternion algebra*. We move all the knowns to the left of the unknown parameters, changing the moving right-handed quaternions into left-handed quaternions.

$$a\hat{q} + c\hat{q}^* = -\hat{b} \tag{1.4}$$
$$c^{*\prime}\hat{q} + a^{*\prime}\hat{q}^* = -\hat{b}^* \tag{1.5}$$

So here, the right-handed quaternions, $c^*, a^* \in \mathbb{H}_R$, become left-handed quaternions, $c^{*\prime}, a^{*\prime} \in \mathbb{H}_L$, all other parameters remaining unchanged. Multiplying the 1st equation by, $a^{*\prime}$, and the 2nd equation by, c, remembering that the left-handed and right-handed quaternions now commute with each other, we get,

$$aa^{*\prime}\hat{q} + ca^{*\prime}\hat{q}^* = -a^{*\prime}\hat{b} \tag{1.6}$$
$$cc^{*\prime}\hat{q} + ca^{*\prime}\hat{q}^* = -c\hat{b}^* \tag{1.7}$$

Subtracting the lower equation from the upper one, we eliminate the term in, $\hat{q}^*$, and obtain the equation,

$$(aa^{*\prime} - cc^{*\prime})\hat{q} = (-a^{*\prime}\hat{b} + c\hat{b}^*) \tag{1.8}$$

To solve this equation, we apply the conjugated cube method, $(h^{*R}h)^{*L}h^{*R}$, where, h, is the two-term factor, $h = (aa^{*\prime} - cc^{*\prime})$, and thus, first multiply both sides of the equation by the right-conjugate factor, $(aa^{*\prime} - cc^{*\prime})^{*R}$,

$$(aa^{*\prime} - cc^{*\prime})^{*R}(aa^{*\prime} - cc^{*\prime})\hat{q} = (aa^{*\prime} - cc^{*\prime})^{*R}(-a^{*\prime}\hat{b} + c\hat{b}^*) \tag{1.9}$$
$$(a^*a^{*\prime} - c^*c^{*\prime})(aa^{*\prime} - cc^{*\prime})\hat{q} = (a^*a^{*\prime} - c^*c^{*\prime})(-a^{*\prime}\hat{b} + c\hat{b}^*) \tag{1.10}$$

---

*Alumnus of the Physics Department of Columbia University, NY.
†Electronic address: pmj29@columbia.edu
[1] see APPENDIX B for discussion on special notation used in this paper, like $\mathbb{H}_R$

obtaining,

$$(|a|^2(a^{*\prime})^2 - a^*ca^{*\prime}c^{*\prime} - c^*ac^{*\prime}a^{*\prime} + |c|^2(c^{*\prime})^2)\hat{q} \quad = \quad (a^*a^{*\prime} - c^*c^{*\prime})(-a^{*\prime}\hat{b} + c\hat{b}^*) \tag{1.11}$$

completing the procedure, we multipy by the conjugated square, $(h^{*R}h)^{*L}$, thus both sides of the equation are now multiplied by the left-conjugate factor, $(|a|^2(a^{*\prime})^2 - a^*ca^{*\prime}c^{*\prime} - c^*ac^{*\prime}a^{*\prime} + |c|^2(c^{*\prime})^2)^{*L}$, yielding,

$$\begin{aligned}(|a|^2(a')^2 - a^*cc'a' - c^*aa'c' + |c|^2(c')^2) \cdot (|a|^2(a^{*\prime})^2 - a^*ca^{*\prime}c^{*\prime} - c^*ac^{*\prime}a^{*\prime} + |c|^2(c^{*\prime})^2)\hat{q} \\ = (|a|^2(a')^2 - a^*cc'a' - c^*aa'c' + |c|^2(c')^2) \cdot (a^*a^{*\prime} - c^*c^{*\prime}) \cdot (-a^{*\prime}\hat{b} + c\hat{b}^*)\end{aligned} \tag{1.12}$$

multiplying out and rearranging,

$$\hat{q} = \left(\begin{array}{l} |a|^2(a^*aa^*a' - a^*ca^*c' - c^*a'c'a^{*\prime} - c^*a'a'c^{*\prime}) \\ \frac{+|c|^2(a^*c'c'a^{*\prime} + a^*c'a'c^{*\prime} + c^*ac^*a' - c^*cc^*c')}{(|a|^2 - |c|^2)^2((|a|^2 + |c|^2)^2 - ((ac^*) + (ac^*)^*)^2)} \end{array} \right) \cdot (-a^{*\prime}\hat{b} + c\hat{b}^*) \tag{1.13}$$

then, simplyfing and moving the left-hand quarternions over to the right of the, $\hat{b}$ and $\hat{b}^*$, parameters, removing the carets, $\hat{}$, [see APPENDIX A for step-by-step derivations] we obtain the final solution for, q, viz.,

$$q = \frac{(a^*aa^* - c^*ac^*)(cb^*a - ba^*a) + (c^*cc^* - a^*ca^*)(cb^*c - ba^*c)}{(|a|^2 - |c|^2)((|a|^2 + |c|^2)^2 - ((ac^*) + (ac^*)^*)^2)} \tag{1.14}$$

This is the "*unit magnitude*" solution to the given quadratic equation, for the special case where the values of the parameters, a, b, c, allow a solution with unit quaterion, $|q| = 1$, to exist. We can now use this special solution to find the other solutions to the quadratic equation, by applying a simple *scale change* to the unknown variable. In particular, we now return to consider the given quadratic equation. We no longer assume that the known parameters have any particular values, such as enabling a unit quaternion solution to exist. Rather, given any values of, a, b, c, we only assume that there exists some non-zero solution, q. We then write this solution as, $q = \lambda p$, where, p, is a unit quaternion, $|p| = 1$, and, $\lambda \in \mathbb{R}, \lambda > 0$, is some real scalar value that represents the magnitude of, q.

Putting, $q = \lambda p$, into the given quadratic equation, results in a similar quadratic equation, this time in the unit quaternion variable, p,

$$\lambda^2 ap^2 + \lambda bp + c = 0 \tag{1.15}$$

Since this is now a quadratic equation with unit quaternion solution, p, we simply replace the known parameters, $a \to \lambda^2 a$, $b \to \lambda b$, $c \to c$, in the *unit magnitude* solution formula, and we immediately find,

$$p = \frac{(\lambda^6|a|^2a^* - \lambda^2c^*ac^*)(\lambda^3cb^*a - \lambda^5ba^*a) + (|c|^2c^* - \lambda^4a^*ca^*)(\lambda cb^*c - \lambda^3ba^*c)}{(\lambda^4|a|^2 - |c|^2)((\lambda^4|a|^2 + |c|^2)^2 - \lambda^4((ac^*) + (ac^*)^*)^2)} \tag{1.16}$$

By considering, p as a function of λ, and solving for λ in the constraint equation, $|p(\lambda)| = 1$, or equivalently,

$$p(\lambda)p(\lambda)^* = 1 \tag{1.17}$$

we can then write the solution for the original quaternion unknown, q, in terms of each λ found to satisfy this constraint equation. Then, for a given, λ, the q, becomes,

$$\begin{aligned} q \quad &= \quad \lambda \cdot \frac{(\lambda^6|a|^2a^* - \lambda^2c^*ac^*)(\lambda^3cb^*a - \lambda^5ba^*a) + (|c|^2c^* - \lambda^4a^*ca^*)(\lambda cb^*c - \lambda^3ba^*c)}{(\lambda^4|a|^2 - |c|^2)((\lambda^4|a|^2 + |c|^2)^2 - \lambda^4((ac^*) + (ac^*)^*)^2)} \\ &= \quad Tq \cdot Uq \end{aligned} \tag{1.18}$$

So, essentially, we have replaced the more complicated quaternion problem in the 4-dimensional space domain, with a scalar problem in a much simpler 1-dimensional space domain. We have a closed form solution to the quaternion quadratic equation, that only requires known magnitudes to determine the directional units.

Typically, we would use numerical methods to find the possible magnitudes, λ_1 and λ_2 , and then plug these magnitudes into the closed form formula to find the corresponding complete quaternions. The closed form solution formula allows us to find the unknown directions from any known magnitudes. The problem of finding the *magnitudes* is "separated" from the problem of finding the *directions*. Thus, the most general method for solving the given quaternion quadratic above involves this *two-step* procedure to find the solutions.

While it is possible to solve some quaternion quadratic equations, with a *one-step* method, that finds both "*magnitude* and *direction*" together, for each q, the *one-step* method does not generalize well to other quadratic equations in quaternions. Instead, this method of separating magnitude from direction has a much wider application. This is in contrast to the situation in complex variable algebra, where a general closed form formula exists, $z = (-b \pm \sqrt{b^2 - 4ac})/(2a)$, to simultaneously determine both *amplitude* and *phase* (equivalent of *magnitude* and *direction* of quaternions) of the complex number solutions for the corresponding quadratic equation, $az^2 + bz + c = 0$, over the complex field, $z \in \mathbb{C}$.

It is somewhat odd, that it is possible to separate the problem of finding *magnitudes* from the problem of finding *directions*. Even stranger, that this *separation is required* to make the method general. But we should note that Sir W. R. Hamilton did recognize this as a general principle, necessary for solving the problems he considered, and made this separation explicit, by introducing his tensor and versor operators, T and U, writing a quaternion as the product of these two, $q = (Tq)(Uq)$. What we call *magnitude*, Hamilton called *Tensor*, and wrote it, Tq, and what we call *direction* or *unit quaternion*, Hamilton called *versor*, and wrote it, Uq. This was one of two typical ways of *separating* the parts of a quaternion, the other being the breakout into *scalar* and *vector*, which he wrote, $q = Sq + Vq$.

Example in Complex Numbers

Let us consider the special case where the coefficients, a, b, c, are restricted to some particular plane through the origin in $\mathbb{H}_R$, that contains the scalar dimension. This means that the known parameters are effectively complex numbers. But, they are in a complex number subspace of the quaternions. Note, we're not replacing the real field with a complex field, as done when introducing *bi-quaternions*. We're still dealing with quaternions over the field of real numbers. We're only restricting the parameters to the form, $x + \iota y \in \mathbb{H}_R, x, y \in \mathbb{R}$, with a general imaginary unit, $\iota = n_1 i + n_2 j + n_3 k$, $n_1, n_2, n_3 \in \mathbb{R}, (n_1^2 + n_2^2 + n_3^3) = 1$, where the four units, $1, i, j, k \in \mathbb{H}_R, ijk = -1$, are the usual basis numbers of the 4-dim right-handed quaternion space.

With this criteria established, the essential feature of importance now is that the known parameters, a, b, c, *commute* with each other. This simplifies the formulas for the *unit magnitude* solution and the constraint, $|p(\lambda)| = 1$.

$$
\begin{aligned}
p &= \frac{(\lambda^6 aa^*a^* - \lambda^2 c^*ac^*)(\lambda^3 cb^*a - \lambda^5 ba^*a) + (cc^*c^* - \lambda^4 a^*ca^*)(\lambda cb^*c - \lambda^3 ba^*c)}{(\lambda^4|a|^2 - |c|^2)((\lambda^4|a|^2 + |c|^2)^2 - \lambda^4((ac^*) + (ac^*)^*)^2)} \\
&= \frac{\lambda^5 aa(\lambda^4 a^*a^* - c^*c^*)(cb^* - \lambda^2 ba^*) + \lambda cc(c^*c^* - \lambda^4 a^*a^*)(cb^* - \lambda^2 ba^*)}{(\lambda^4|a|^2 - |c|^2)((\lambda^4|a|^2 + |c|^2)^2 - \lambda^4((ac^*) + (ac^*)^*)^2)} \\
&= \frac{\lambda \cdot (\lambda^4 aa - cc)(\lambda^4 a^*a^* - c^*c^*)}{(\lambda^4|a|^2 - |c|^2)((\lambda^4|a|^2 + |c|^2)^2 - \lambda^4((ac^*) + (ac^*)^*)^2)} \cdot (cb^* - \lambda^2 ba^*) \\
&= \frac{\lambda \cdot (cb^* - \lambda^2 ba^*)}{(\lambda^4|a|^2 - |c|^2)}
\end{aligned}
\tag{1.19}
$$

Note that in the special case when, a, b, c, are real number parameters, this formula further reduces to,

$$p = \frac{\lambda \cdot (c - \lambda^2 a)b}{(\lambda^4 a^2 - c^2)} = \frac{-\lambda b}{(\lambda^2 a + c)} \tag{1.20}$$

Given that, $\lambda, a, b, c \in \mathbb{R}$, are all real now, the *unit quaternion*, p, is real also, and thus, the constraint, $|p(\lambda)| = 1$, simply requires that, $p = \pm 1$. Setting, $p = +1$, and then, $p = -1$, gives us two equations for determining λ.

$$p = +1, \quad a\lambda^2 + b\lambda + c = 0 \quad \rightarrow \lambda = (-b \pm \sqrt{b^2 - 4ac})/(2a) \quad \rightarrow q = \lambda \cdot p = \lambda \cdot (+1) \tag{1.21}$$

$$p = -1, \quad a\lambda^2 - b\lambda + c = 0 \quad \rightarrow \lambda = (+b \pm \sqrt{b^2 - 4ac})/(2a) \quad \rightarrow q = \lambda \cdot p = \lambda \cdot (-1) \tag{1.22}$$

Of the 4 available roots for λ, two are negative and two are positive, whenever real roots exist; only the positive two roots, $\lambda > 0$, are used to construct the solution. The sign of the solution, q, is established by the direction unit, p, corresponding to the λ for each positive magnitude solution found. This method of solution, that separates the *magnitude* from *direction*, thus requiring a *two-step* procedure in the working out, is obviously not as efficient as the method of *completing the square* typically used for solving quadratic equations in complex numbers or reals. Nevertheless, it is instructive to see how the quaternion formula transforms when we regain *commutation* by restricting to complex numbers, and then regain *ordering* by further restriction to the reals. One result of this transformation is that we obtain a formula for the *direction* of the complex number, i.e. its *phase angle*, ϕ, once we're given the *magnitude*, i.e. *amplitude* or *modulus*, λ.

$$p = \frac{\lambda \cdot (cb^* - \lambda^2 ba^*)}{(\lambda^4 |a|^2 - |c|^2)} = e^{\iota\phi} \tag{1.23}$$

The unknown *phase*, $e^{\iota\phi}$, is eliminated by taking the complex conjugate, and multiplying to obtain, $pp^* = 1$, which is then a scalar equation in just the magnitude, λ.

$$pp^* = \frac{\lambda^2 \cdot (\lambda^4 |ab|^2 - 2\lambda^2 Re[b^2(ac)^*] + |bc|^2)}{(\lambda^4 |a|^2 - |c|^2)^2} = 1 \tag{1.24}$$

Once we've found suitable magnitudes, λ_1 and λ_2, we plug them into the equation for the phase, to determine the phase angle that corresponds with each magnitude.

But, almost immediately, we percieve potential issues with the method. The function that maps magnitude to direction result, $\lambda \rightarrow p$, can only produce one direction, p, for each magnitude, λ. There are no square-roots here, to give us multiple outputs for a given input. The solution is expressed in terms of a ratio of integer exponent polynomials in λ. So, the first obvious question is, how do we find two unique solutions with same magnitude that have two different phases? When we plug in a λ_1, we get out one phase, $e^{i\phi_1}$. To get a different phase, $e^{i\phi_2}$, we need a different λ_2. The second question is, can two different magnitudes have the same phase? The third question is, what happens to the solution when the denominator is zero? These issues don't exist in the method of *Completing the Square*, so why do they exist here? So, to understand the problem better, let's construct a quadratic equation with known roots, z_1 and z_2, in the complex number domain.

$$(z - z_1)(z - z_2) = 0 \tag{1.25}$$

$$z^2 - (z_1 + z_2)z + z_1 z_2 = 0 \tag{1.26}$$

$$a = 1, \;\; b = -(z_1 + z_2), \;\; c = z_1 z_2 \tag{1.27}$$

plugging in the values for, a, b, c, into, $p(\lambda)$, we have,

$$p = \frac{\lambda \cdot (cb^* - \lambda^2 ba^*)}{(\lambda^4 |a|^2 - |c|^2)} = \frac{\lambda \cdot (-z_1 z_2 (z_1^* + z_2^*) + \lambda^2 (z_1 + z_2))}{(\lambda^4 - |z_1 z_2|^2)} = \frac{\lambda \cdot (\lambda^2 (z_1 + z_2) - (z_1 |z_2|^2 + z_2 |z_1|^2))}{(\lambda^4 - |z_1 z_2|^2)} \tag{1.28}$$

Let's now consider the following special cases.

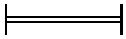
CASE STUDIES

`Case 1 - Same magnitude, different phases:`
$z_1 = m \cdot u_1, z_2 = m \cdot u_2;\ \ m \in \mathbb{R}, m > 0;\ u_1, u_2 \in \mathbb{C},\ |u_1| = 1, |u_2| = 1, u_1 \neq u_2.$

`Case 2 - Same phase, different magnitudes:`
$z_1 = m \cdot u, z_2 = n \cdot u;\ \ m, n \in \mathbb{R}, m, n > 0, m \neq n;\ u \in \mathbb{C},\ |u| = 1.$

`Case 3 - Different magnitudes, different phases:`
$z_1 = m \cdot u_1, z_2 = n \cdot u_2;\ \ m, n \in \mathbb{R}, m, n > 0, m \neq n;\ u_1, u_2 \in \mathbb{C},\ |u_1| = 1, |u_2| = 1, u_1 \neq u_2.$

`Case 4 - Vanishing Denominator:`
$|\lambda^4|a|^2 - |c|^2| = 0$



§:|:

`Case 1 - Same magnitude, different phases:`
$z_1 = m \cdot u_1, z_2 = m \cdot u_2, m \in \mathbb{R}, m > 0,\ u_1, u_2 \in \mathbb{C},\ |u_1| = 1, |u_2| = 1, u_1 \neq u_2.$

$$p = \frac{m\lambda \cdot (\lambda^2 - m^2)(u_1 + u_2)}{(\lambda^4 - m^4)} = \frac{m\lambda(u_1 + u_2)}{(\lambda^2 + m^2)} \tag{1.29}$$

The requirement that, $|p(\lambda)| = 1$, means that,

$$|m\lambda(u_1 + u_2)| = |\lambda^2 + m^2| \tag{1.30}$$

But we know that, $|u_1 + u_2| \leq |u_1| + |u_2| = 1 + 1 = 2$, and also, $|\lambda^2 + m^2| - |2m\lambda| = |\lambda^2 + m^2 - 2m\lambda|$, since, $\lambda > 0, m > 0$. But, then, $|\lambda^2 + m^2| - |2m\lambda| \geq 0$, i.e., $|\lambda^2 + m^2| \geq |2m\lambda|$. Hence,

$$|m\lambda(u_1 + u_2)| \leq |2m\lambda| \leq |\lambda^2 + m^2| \tag{1.31}$$

$$\text{thus,} \qquad |p| \ = \ \left|\frac{m\lambda(u_1 + u_2)}{\lambda^2 + m^2}\right| \ = \ \left|\frac{m\lambda}{\lambda^2 + m^2}\right| \sqrt{2 + u_1 u_2^* + u_2 u_1^*} \ \leq \ 1 \tag{1.32}$$

The solution method is attempting to make the match, $\lambda = m$, to satisfy, $|p| = 1$. This is successful when the unit directions, i.e. here the phases of the complex numbers, are the same, $u_1 = u_2$. But when the directions are not aligned, then there is no value for λ that can make the equality match, $|p| = 1$. However, notice that we can still find the magnitude, $\lambda = m$, if we change the constraint requirement. If there's no λ that satisfies, $|p| = 1$, yet, $|p| \not\equiv 0$, we just search for the λ that maximizes $|p|$. Since, the function, $|p(\lambda)| \to 0$, as $\lambda \to \pm\infty$, it must have a maximum somewhere. Moreover that maximum must occur in the right half line, $\lambda \in [0, \infty)$, so by changing the constraint requirement, from, $|p(\lambda)| = 1$, to, $\max(p(\lambda))$, we are still able to find, $\lambda = m$. The problem now, however, is that the, $p(m)$, for this case, is an average of the two units. In this case, our '*provisional*' unit, p, is not an actual unit, $|p| = 1$, since the simple arithmetic average of two different unit directions is not itself of unit magnitude.

$$p(\lambda = m) = \frac{m^2(u_1 + u_2)}{(m^2 + m^2)} = \frac{1}{2}(u_1 + u_2) \tag{1.33}$$

However, now that we have an equation for the units, u_1, u_2, we can take this equation, and conjugate it, to again obtain a system of linear equations,

$$u_1 + u_2 \quad = 2p \tag{1.34}$$
$$u_1^* + u_2^* \quad = 2p^* \tag{1.35}$$

Here, $p = p(\lambda = m)$, gives the maximum of the function, $|p(\lambda)|$, but its norm is less than unity, $|p| < 1$. While, u_1 and u_2, are true unit directions, $|u_1| = 1$ and $|u_2| = 1$, so that, $u_1^* = 1/u_1$ and $u_2^* = 1/u_2$. So, from the first equation,

we have, $u_2 = 2p - u_1$, which lets us write, $u_2^* = 1/u_2 = 1/(2p - u_1)$. Plugging this into the second equation, where we also replace, $u_1^* = 1/u_1$, and rearranging, we obtain a new quadratic equation, with unknown u_1. Since the pair of simultaneous equations is completely symmetric in the exchange of the two units, u_1 and u_2, we also get a matching equation in the unknown u_2. These two unit directions are then just the two roots of the same quadratic, which is,

$$p^* u^2 - 2|p|^2 u + p = 0 \tag{1.36}$$

The two roots of this quadratic, are the two unit directions, $u = u_1$, and, $u = u_2$. The solution is ($\iota = \sqrt{-1}$),

$$\begin{array}{lll} u_1 = & p \cdot (\ 1 + \iota\sqrt{|p|^{-2} - 1}\ ) & ; \quad u_2 = \quad p \cdot (\ 1 - \iota\sqrt{|p|^{-2} - 1}\ ) \\ z_1 = m \cdot p \cdot (\ 1 + \iota\sqrt{|p|^{-2} - 1}\ ) & ; & z_2 = m \cdot p \cdot (\ 1 - \iota\sqrt{|p|^{-2} - 1}\ ) \end{array} \tag{1.37}$$

We had to resort back to the usual method of *Completing the Square*, to extract the unit directions, u_1 and u_2, for this last part. However, it shows that, *in principle*, the solution, at least in the case of complex numbers, can be recovered from just knowing the magnitude, $\lambda = m$, and *the average* of the units, $p = 1/2 \cdot (u_1 + u_2)$, two things which the *two-step* method can readily find, in the special problematic case where the magnitude is the same, but the directions differ.

`A third step!:`
In the case of complex numbers then, once the "*two-step*" method fails to find a solution with the usual constraint, $|p(\lambda)| = 1$, we modify the constraint requirement to finding the "`max` $p|(\lambda)|$," instead. Then, once we've found the degenerate magnitude, $\lambda = m$, and average direction, $p = (u_1 + u_2)/2$, with, $|p| < 1$, we can just "fix up" the direction, by multiplying the average direction by the magical conjugate pair of factors, $(1 + \iota\sqrt{|p|^{-2} - 1})$, and $(1 - \iota\sqrt{|p|^{-2} - 1})$, obtained from our *secret knowledge of completing the square*, and we immediately recover our two distinct unit directions, u_1 and u_2. Of course, when the different unit directions are of the special case, $u_1 = -u_2$, our provisional unit direction vanishes everywhere, $p \equiv 0$ (actually, more detailed analysis shows then, $p = 0/0$, so, p is just undetemined here by the two-step method), so we can't even find, $\lambda = m$, much less determine the two, u_1 and u_2, units. But, that only happens when, $b = 0$, since, $b = -(z_1 + z_2)$, and we've excluded the quadratic equations with such vanishing, b, from our two-step method. A simpler one-step method solves those equations, anyway.

`Case 2 - Same phase, different magnitudes:`
$z_1 = m \cdot u, z_2 = n \cdot u;\ \ m, n \in \mathbb{R}, m, n > 0, m \neq n;\ u \in \mathbb{C},\ |u| = 1.$

$$p = \frac{(m+n)\lambda \cdot (\lambda^2 - mn)u}{(\lambda^4 - m^2 n^2)} = \frac{(m+n)\lambda u}{(\lambda^2 + mn)} \tag{1.38}$$

The requirement that, $|p(\lambda)| = 1$, means that,

$$|(m+n)\lambda u| = |(m+n)\lambda|\ |u| = |(\lambda^2 + mn)| \tag{1.39}$$

Here, $|u| = 1$, and thus both, $\lambda = m$, and, $\lambda = n$, easily satisfy the constraint, and either magnitude yields the same direction unit, u,

$$\lambda = m, \qquad |(m+n)m| = |(m^2 + mn)|\ , \qquad p = \frac{(m+n)mu}{(m^2 + mn)} = u \tag{1.40}$$

$$\lambda = n, \qquad |(m+n)n| = |(n^2 + mn)|\ , \qquad p = \frac{(m+n)nu}{(n^2 + mn)} = u \tag{1.41}$$

Therefore, there is no inherent problem with the two-step method finding the solutions in this case. In principle, the problem is completely solvable by this method.

`Case 3 - Different magnitudes, different phases:`
$z_1 = m \cdot u_1, z_2 = n \cdot u_2;\ \ m, n \in \mathbb{R}, m, n > 0, m \neq n;\ u_1, u_2 \in \mathbb{C},\ |u_1| = 1, |u_2| = 1, u_1 \neq u_2.$

$$p = \frac{m\lambda \cdot (\lambda^2 - n^2)u_1 + n\lambda \cdot (\lambda^2 - m^2)u_2}{(\lambda^4 - m^2n^2)} \tag{1.42}$$

The requirement that, $|p(\lambda)| = 1$, means that,

$$|m\lambda \cdot (\lambda^2 - n^2)u_1 + n\lambda \cdot (\lambda^2 - m^2)u_2| = |(\lambda^4 - m^2n^2)| \tag{1.43}$$

The method tries to match, $\lambda = m$, so, to see if this is always possible, we replace λ,

$$\lambda = m, \tag{1.44}$$
$$|mm \cdot (m^2 - n^2)u_1 + nm \cdot (m^2 - m^2)u_2| = |(m^4 - m^2n^2)| \tag{1.45}$$
$$|mm \cdot (m^2 - n^2)u_1 + 0| = |m^2(m^2 - n^2)| \tag{1.46}$$
$$|mm||(m^2 - n^2)||u_1| = |m^2||(m^2 - n^2)| \tag{1.47}$$

Match succeeds, since, $|u_1| = 1$, and the method finds the magnitude, $\lambda = m$, and direction unit,

$$p = \frac{mm \cdot (m^2 - n^2)u_1 + nm \cdot (m^2 - m^2)u_2}{(m^4 - m^2n^2)} = \frac{mm \cdot (m^2 - n^2)u_1}{m^2(m^2 - n^2)} = u_1 \tag{1.48}$$

Then the method tries to match, $\lambda = n$, so, to see if this is always possible, we replace λ,

$$\lambda = n, \tag{1.49}$$
$$|mn \cdot (n^2 - n^2)u_1 + nn \cdot (n^2 - m^2)u_2| = |(n^4 - m^2n^2)| \tag{1.50}$$
$$|0 + nn \cdot (n^2 - m^2)u_2| = |n^2(n^2 - m^2)| \tag{1.51}$$
$$|nn||(n^2 - m^2)||u_2| = |n^2||(n^2 - m^2)| \tag{1.52}$$

Match succeeds, since, $|u_2| = 1$, and the method finds the magnitude, $\lambda = n$, and direction unit,

$$p = \frac{mn \cdot (n^2 - n^2)u_1 + nn \cdot (n^2 - m^2)u_2}{(n^4 - m^2n^2)} = \frac{nn \cdot (n^2 - m^2)u_2}{n^2(n^2 - m^2)} = u_2 \tag{1.53}$$

So, there are no inherent problems with this case. The two-step method should, in principle, be able to find both solutions.

`Case 4 - Vanishing Denominator:`
$|\lambda^4|a|^2 - |c|^2| = 0$

The denominator vanishes when, $\lambda = \sqrt{|c|/|a|},\ \ |a| \neq 0$; and from the previous cases we notice that this is where, $(\lambda^4 - m^2n^2) = 0$, where, $\lambda = m$ and $\lambda = n$, are the two magnitudes. So, we notice, the vanishing only occurs at the root we seek, when the magnitudes are the same, $m = n$. Otherwise, the zero is elsewhere, and it doesn't interfere with our ability to find the solutions. The issue, then, becomes important, only for the cases considered where we have *same magnitudes*. In the analysis above, we were able to simplify the numerator and denominator expressions, given the assumed roots, to show that, "*in principle*," solutions are possible. Of course, we don't really have foreknowledge of the solutions, so we can't artfully cancel out the offending zero factors from the numerator and denominator to obtain the results like we did "in theory." Instead, "*in practice*," we use *L'Hôspital's Rule* to overcome the vanishing terms, by differentiating the numerator and denominator, when we find zeros there, replacing, p, with, P;

$$p = \frac{\lambda \cdot (cb^* - \lambda^2 ba^*)}{(\lambda^4|a|^2 - |c|^2)} \qquad \rightarrow \quad P = \frac{(cb^* - 3\lambda^2 ba^*)}{4\lambda^3|a|^2} \tag{1.54}$$

$$p = \frac{m\lambda \cdot (\lambda^2 - n^2)u_1 + n\lambda \cdot (\lambda^2 - m^2)u_2}{(\lambda^4 - m^2n^2)} \qquad \rightarrow \quad P = \frac{m \cdot (3\lambda^2 - n^2)u_1 + n \cdot (3\lambda^2 - m^2)u_2}{4\lambda^3} \tag{1.55}$$

$$p(\lambda = m; m = n) = \frac{0}{0} \qquad \rightarrow \quad P(\lambda = m; m = n) = \frac{1}{2}(u_1 + u_2) \tag{1.56}$$

This enables us to use our '`third step`' technique in the *same magnitudes, different directions* case.

Example in Quaternions (#1)

To prepare to implement the solution in code, we first rearrange the numerator and denominator of the unit magnitude formula, to reduce the number of multiplications required, and to see more clearly where the numerical problems might arise from a vanishing of the denominator. Thus we rewrite, $p(\lambda)$, as,

$$p = \frac{\lambda^5(\lambda^4 a^* a a^* - c^* a c^*)(c b^* - \lambda^2 b a^*)a + \lambda(c^* c c^* - \lambda^4 a^* c a^*)(c b^* - \lambda^2 b a^*)c}{(\lambda^4|a|^2 - |c|^2)\cdot|\lambda^2 a - c|^2\cdot|\lambda^2 a + c|^2} \tag{1.57}$$

Presumably, $|a| \neq 0$, because it is assumed the equation is quadratic. If, $|c| = 0$, then the above reduces to,

$$p = \frac{(-a^* b)}{\lambda|a|^2} \tag{1.58}$$

So, the constraint, $|p| = 1$, gives us, $\lambda = |b|/|a|$. Plugging in this solution for the magnitude, gives the direction, $p = (-a^* b)/(|ab|)$, and thus the complete solution is, $q = \lambda \cdot p(\lambda) = |b|/|a| \cdot (-a^* b)/(|ab|) = (-a^*/|a|^2)b = -a^{-1}b$. But, when $|c| = 0$, the quadratic equation also simplifies to, $aq^2 + bq = 0$, which factors easily into, $(aq + b)q = 0$, yielding the two solutions, $q = 0$, and, $q = -a^{-1}b$. So, there's no need for the *two-step* method there. Notice, however, that the zero solution, $q = 0$, is not obtained by the *two-step* method! We can't find, $q = 0$, because the whole premise of the method is that we have a non-zero solution, q, which can be scaled, to produce a non-zero quaternion with "unit magnitude," p, in order to write, $q = \lambda p$, to effect the method. Note that, if, $|a| = 0$, but, $|b| \neq 0,\ |c| \neq 0$, then although the quadratic collapses to a simple linear equation, $bq + c = 0$, this *two-step* method finds the solution, $q = -b^{-1}c$, just fine. As long as the solution, q, is non-zero, the method works.[2]

Therefore, we ignore the case where, $|a| = |c| = 0$, which causes both numerator and denominator to vanish, because such an equation is not quadratic. There are then 2 special cases left to consider; the first, when, $\exists\gamma \in \mathbb{R}\ s.t.\ c = \gamma a$, so the two parameters are proportional to each other, and the second when a and c are independent, $c \neq \gamma a,\ \forall\gamma \in \mathbb{R}$,

`Case 1:` $\exists\gamma \in \mathbb{R}\ ,\ \gamma \neq 0,\ s.t.,\ c = \gamma a.$

$$p = \frac{\lambda(\lambda^4 - \gamma^2)(\lambda^4 - \gamma^2)}{(\lambda^4 - |\gamma|^2)\cdot|\lambda^2 - \gamma|^2\cdot|\lambda^2 + \gamma|^2}\cdot\frac{(\gamma b^* a - \lambda^2 a^* b)}{|a|^2} \tag{1.59}$$

$$= \frac{\lambda}{(\lambda^4 - |\gamma|^2)}\cdot\frac{(\gamma b^* a - \lambda^2 a^* b)}{|a|^2} \tag{1.60}$$

So here, zeros in the denominator occur only when, $\lambda^2 = |\gamma| = |c|/|a|$, that is to say, $\lambda = \pm\sqrt{|c|/|a|}$.

`Case 2:` $c \neq \gamma a,\ \forall\gamma \in \mathbb{R}$.
Here, neither of the two factors, $|\lambda^2 a - c|^2$, nor, $|\lambda^2 a + c|^2$, are 0 for any value of λ. So, the only source of vanishing denominator is the term, $(\lambda^4|a|^2 - |c|^2) = 0$, which happens at exactly one point in the domain, $[\ 0, \infty)$, at, $\lambda = \sqrt{|c|/|a|}$. A corresponding zero point exists in the negative half line, $(-\infty, 0\ ]$, but, $\lambda > 0$, by design, so we don't have to look there; although when we generate plots, it's useful to see the whole range for the function, even where we're not specifically looking for λ, since we then observe the minima at the vertical axis, where, $\lambda = 0$, a bit better.

All things considered, then, we expect that in searching for solutions to the magnitude, λ, using this formula, to have numerical problems at just one point in the domain of interest, $\lambda \in [\ 0, +\infty)$, a point we know of in advance, being determined by the absolute ratio of the coefficients, $|c|/|a|$. Looking in the whole line, $\lambda \in (-\infty, +\infty)$, there are two points to consider, given that the other half line has a symmetrically placed point where things should blow up too. So, we mark these two points in our search,

$$\lambda\ =\ -\sqrt{|c|/|a|} \quad ; \quad \lambda =\ +\sqrt{|c|/|a|} \tag{1.61}$$

---

[2] When $q = 0$ is an obvious solution by simple inspection, the quadratic 'may sometimes' be easily transformed into a linear equation. Although this equation is straightforward, quadratic equations of the form, $a_1 q a_2 q a_3 + b_1 q b_2 = 0$, are a bit more challenging.

Apart from the two spike points, consider the behavior of the provisional unit direction, $p(\lambda)$, and the square norm functions, $|p|^2 = p(\lambda)p(\lambda)^*$, as, $\lambda \to \pm\infty$, and, $\lambda \to 0$.

$$|a| \neq 0,\ |b| \neq 0,\ |c| \neq 0 \qquad : \tag{1.62}$$

$$\lambda \to \pm\infty \quad , \qquad p = -\frac{a^*b}{\lambda|a|^2} \ \to 0 \quad ; \quad pp^* = \frac{|b|^2}{\lambda^2|a|^2} \ \to 0 \tag{1.63}$$

$$\lambda \to 0 \quad , \qquad p = -\frac{\lambda b^*c}{|c|^2} \ \to 0 \quad ; \quad pp^* = \frac{\lambda^2|b|^2}{|c|^2} \ \to 0 \tag{1.64}$$

We have a non-negative value function, $|p(\lambda)|^2 \geq 0$, that goes to 0 at the two ends of its range, over the domain, $\lambda \in [\ 0, +\infty)$, so that the function must attain a maximum somewhere within the range. If that max is > 1, i.e. $|p(\lambda)|^2_{max} > 1$, then the equation, $pp^* = 1$, has exactly two solutions for the magnitude, λ_1 and λ_2, that are distinct and different in value from each other. Moreover, because the spike point is the location of the max, we have,

$$0 < \lambda_1 < +\sqrt{|c|/|a|}, \qquad \text{and,} \qquad +\sqrt{|c|/|a|} < \lambda_2 < +\infty \tag{1.65}$$

$$p(\lambda_1)p(\lambda_1)^* = 1 \qquad , \qquad p(\lambda_2)p(\lambda_2)^* = 1 \tag{1.66}$$

With this bit of analysis, we have a good sense of where to look to find the solutions. Much of this analysis depends on the parameter, b, being non-zero. So, let's take a look at this parameter.

When, $b = b^*,\ |b| \neq 0$, the parameter, b, is real, and the formula for, p, simplifies to,

$$p = b \cdot \frac{\lambda^5(\lambda^4 a^*aa^* - c^*ac^*)(c - \lambda^2 a^*)a + \lambda(c^*cc^* - \lambda^4 a^*ca^*)(c - \lambda^2 a^*)c}{(\lambda^4|a|^2 - |c|^2) \cdot |\lambda^2 a - c|^2 \cdot |\lambda^2 a + c|^2} \tag{1.67}$$

But, when, $b = 0$, there are no solutions obtainable by the two-step solution. The method can't find, p, since, $p = 0$, is not a valid option (actually, on closer inspection, $p = 0/0$, here, as seen next, so p is undetermined). With, $b \in \mathbb{R},\ b \neq 0$, the method works just fine. But, with, $b = 0$, the quadratic equation transforms to,

$$aq^2 + c \ = 0 \tag{1.68}$$

$$q^2 \ = -\frac{a^*c}{|a|^2} \tag{1.69}$$

Putting, $q = \lambda p$, with, $|p| = 1, \lambda \in \mathbb{R},\ \lambda > 0$, as before, yields, $\lambda^2 = |q|^2 = |c|/|a|$, so that, $\lambda = \sqrt{|c|/|a|}$. But, this is the exact λ point where our two-step formula blows up! So we can't use it. Here, this gives us, $p^2 = q^2/\lambda^2$, or,

$$p^2 = -\frac{a^*c}{|a||c|} \tag{1.70}$$

If, $a^*c \in \mathbb{R}$, and, $a^*c > 0$, then, $p^2 = -1$, and so there are an infinity of solutions, $p = (n_1 i + n_2 j + n_3 k),\ n_1, n_2, n_3 \in \mathbb{R},\ n_1^2 + n_2^2 + n_3^2 = 1$. If, $a^*c \in \mathbb{R}$, and, $a^*c < 0$, then, $p^2 = +1$, and so there are just two solutions, $p = \pm 1$. If, $a^*c \notin \mathbb{R}$, and, $S(a^*c) = 0$, then, there are two solutions, $p = \pm(1 - a^*c/(|a||c|))/\sqrt{2}$. And finally, if, $a^*c \notin \mathbb{R}$, and, $S(a^*c) \neq 0$, there are two solutions, $p = \pm(1 - a^*c/(|a||c|))/(\sqrt{(}2 - (a^*c + (a^*c)^*)/|a^*c|))$. These several special cases are not general, so from here on, we consider only, $b \neq 0$, cases. [See APPENDIX C for the, $b = 0$, derivations.]

TESTING 1

In order to perform quaternion calculations for testing, using one of the popular mathematical software tools like MATLAB, it helps to have a convenient representation that can be easily interpreted by the software. One convenient representation is matrices with complex number entries, another is with matrices over reals. The complex number version is more compact, and we'll consider that here.

`A` 2×2 `Representation`.

A quaternion can be written as a 2×2 matrix over complex numbers. Two such popular representations were discussed in our previous paper on *Quatro-Quaternions* ([PJ3]):

CAYLEY-DICKSON (I):
$$(w,v)^* \equiv (w^*, -v) \;\;,\;\; A + iB, \qquad \equiv \mathbf{LH} \tag{1.71}$$
$$(w_1,v_1)(w_2,v_2) = (w_1w_2 - v_2v_1^* \;,\; w_1^*v_2 + w_2v_1) \tag{1.72}$$

$$q = \begin{pmatrix} w & -v^* \\ v & w^* \end{pmatrix} \quad , \quad q^* = \begin{pmatrix} w^* & v^* \\ -v & w \end{pmatrix} \quad , \quad w, v \in \mathbb{C}, \tag{1.73}$$
$$q = t + xi + yj + zk; \;\; \underline{\boldsymbol{ijk = +1}}; \;\; w = t + \iota x, v = y + \iota z, \;\; \iota^2 = -1 \tag{1.74}$$

CAYLEY-DICKSON (II):
$$(w,v)^* \equiv (w^*, -v) \;\;,\;\; A + Bi, \qquad \equiv \mathbf{RH} \tag{1.75}$$
$$(w_1,v_1)(w_2,v_2) = (w_1w_2 - v_2^*v_1 \;,\; v_2w_1 + v_1w_2^*) \tag{1.76}$$

$$q = \begin{pmatrix} w & v \\ -v^* & w^* \end{pmatrix} \quad , \quad q^* = \begin{pmatrix} w^* & -v \\ v^* & w \end{pmatrix} \quad , \quad w, v \in \mathbb{C}, \; q, q^* \in \mathbb{H}_R \tag{1.77}$$
$$q = t + xi + yj + zk; \;\; \underline{\boldsymbol{ijk = -1}}; \;\; w = t + \iota x, v = y + \iota z, \;\; \iota^2 = -1 \tag{1.78}$$
$$\texttt{in MATLAB this is:} \quad w = t + 1i * x; \; v = y + 1i * z; \; q = [w, v; -v', w']. \tag{1.79}$$

What we'd like to point out here, is that, what we labeled CAYLEY-DICKSON (I), is actually a `left-hand` system, with, $ijk = +1$; and the CAYLEY-DICKSON (II), is a `right-hand` system,[3] with, $ijk = -1$. Here, *quaternion conjugation*, q^*, involves *complex conjugation* on the entries of the 2×2 matrix AND also *transposition* of the matrix. The combined effect of this operation is to conjugate the main diagonal entries and flip the signs on the cross diagonal entries. We have a right-hand and a left-hand pair. We pick the RH to work with, it's all we need, but the question naturally arises, is this LH *the distinguished left-hand*, or just another left-hand number? Let's breakout the units to see.

$$\begin{aligned} \mathbf{RH:} \quad \mathbf{1} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \boldsymbol{i} = \begin{pmatrix} \iota & 0 \\ 0 & -\iota \end{pmatrix} \quad \boldsymbol{j} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \quad \boldsymbol{k} = \begin{pmatrix} 0 & \iota \\ \iota & 0 \end{pmatrix} \\ \mathbf{LH:} \quad \mathbf{1} &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \boldsymbol{i} = \begin{pmatrix} \iota & 0 \\ 0 & -\iota \end{pmatrix} \quad \boldsymbol{j} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \quad \boldsymbol{k} = \begin{pmatrix} 0 & \iota \\ \iota & 0 \end{pmatrix} \end{aligned} \tag{1.80}$$

A quick glance at the units reveals that these two systems are not linearly independent. In fact, the left hand triple is obtained by conjugating just one imaginary from the right hand triple. So, if the RH is, i, j, k , then this corresponding LH is simply, i, j^*, k . That tells us immediately that this left hand doesn't commute with this right hand. As discussed in APPENDIX B, the *distinguished left-hand* must both *co-permute* and then *commute* with the given right hand. Since this doesn't even commute, it can't play the role of representation of the distinguished left hand. As a result, we won't use our special hand transform mark $'$, nor the designation, $\mathbb{H}_L$, to refer to the LH here. We reserve that notation for the distinguished left hand as discussed in the appendix on notation conventions.

So far in our papers, we've noticed three different types of left-hand numbers, all linked in some way with a given right hand: there's the conjugate basis, i^*, j^*, k^*, that inverts 3 units; there's the distinguished left, i', j', k', that commutes with the right; and then there's this new left, i, j^*, k, that inverts just 1 unit. This tells us we must test the left-hand to make sure it is suitable for our purpose before using some representation that just happens to obey the *internal rules*, $ijk = +1$. An appropriate left-hand must interact with the right hand in a special way.

---

[3] A 'left side' i yields the LH quaternions with $A + iB$, while a 'right side' i yields the RH quaternions with $A + Bi$. See previous paper on *Quatro-Quaternions* for discussion on how the two different forms, $A + iB$ and $A + Bi$, with imaginary i on the left and right of B, generate the two versions of the CAYLEY-DICKSON construction.

Having said this, the RH 2×2 matrix representation here is perfect for our purpose of calculating with just the right-hand quaternions alone.

in MATLAB code: w = t + 1i*x; v = y + 1i*z; q = [w,v;-v′,w′]; % q = t + xi + yj + zk.

```
function psolqabc(A,B,C)
global a b c;
% e.g:  A = [ 2, 1, 0, -1 ]; % a = 2 + 1i + 0j - 1k
% e.g:  B = [ -3, 0, 1, 1 ]; % b = -3 + 0i + 1j + 1k
% e.g:  C = [ 1, 0, -1, 1 ]; % c = 1 + 0i - 1j + 1k

a = [(A(1)+1i*A(2)),(A(3)+1i*A(4));-(A(3)+1i*A(4))′,(A(1)+1i*A(2))′];
b = [(B(1)+1i*B(2)),(B(3)+1i*B(4));-(B(3)+1i*B(4))′,(B(1)+1i*B(2))′];
c = [(C(1)+1i*C(2)),(C(3)+1i*C(4));-(C(3)+1i*C(4))′,(C(1)+1i*C(2))′];
end

function up=psolq(r)
global a b c;
p = psolqd(r);
pp = trace(p*p′)/2;
up = pp - 1;
end

function p=psolqd(r)     % ver:  1.0,  (aqq + bq + c = 0)
global a b c;
np = r^5*(r^4*a′*a*a′ - c′*a*c′)*(c*b′ - r^2*b*a′)*a;
np = np + r*(c′*c*c′ - r^4*a′*c*a′)*(c*b′ - r^2*b*a′)*c;
dp = trace((r^4*a*a′ - c*c′)*(r^2*a - c)*(r^2*a - c)′*(r^2*a + c)*(r^2*a + c)′)/2;
p = np/dp;
end

function P=psolqv(r)
global a b c;
p = psolqd(r);
P = [ real(p(1,1)) , imag(p(1,1)) , real(p(1,2)), imag(p(1,2)) ];
end

function V=psolqck(r)
global a b c;
Q = r*psolqv(r);
V = psolqev(Q);
end

function V=psolqev(Q)     % ver:  1.0,  (aqq + bq + c = 0)
global a b c;
q = [(Q(1)+1i*Q(2)),(Q(3)+1i*Q(4));-(Q(3)+1i*Q(4))′,(Q(1)+1i*Q(2))′];
v = a*q*q + b*q + c;
V = [real(v(1,1)) , imag(v(1,1)), real(v(1,2)), imag(v(1,2))];
end

>> global a b c;
>> A = [ 2, 1, 0, -1 ]; % a = 2 + 1i + 0j - 1k
>> B = [ -3, 0, 1, 1 ]; % b = -3 + 0i + 1j + 1k
>> C = [ 1, 0, -1, 1 ]; % c = 1 + 0i - 1j + 1k
>> psolqabc(A,B,C);
>> r1 = fzero(@psolq,[0.5,0.7])
>> P = psolqv(r1)
>> V = psolqck(r1)
```

There are 7 MATLAB functions defined here for finding and testing solutions. This is a very simple and bare bones collection of routines, that provide just the essense of the calculations needed to explore the method. The reader is welcome to modify them to his advantage. We make the three known parameters, a, b, c, global variables. So, we're only dealing with one quaternion quadratic equation, $aq^2 + bq + c = 0$, at a time. Although global is declared in every function, this is really just a reminder. We need global declaration only where variables actually appear in code. We also declare, `global a b c;` at the command prompt in the MATLAB command window, so we can access them there, and get things going.

First, at the command prompt we define three row vector variables, `A, B, C`, containing the values of the known parameters, a, b, c. Then, we call the function, `psolqabc(A,B,C)`, which converts these parameters from simple vectors into the 2×2 complex number matrix format that is actually used in calculations. These 2×2 matrices are our global, `a, b, c`, which are initialized and setup by this function. Everytime we want to change the parameters to another quadratic, we have to call this function again, with the new, `A, B, C`, we've defined. MATLAB uses the ASCII character ′ to refer to "hermitian conjugation," which is just "complex conjugation" when applied to complex numbers, but includes "transposition" also when applied to matrices. This is just what we want. So, don't confuse the mark here with our "hand transformation" operator. MATLAB doesn't know anything about the left hand.

The function, `psolq(r)`, is the workhorse of the collection. It computes the function, $f(\lambda) = p(\lambda)p(\lambda)^* - 1$, providing us with a function whose zeros are the solutions we seek. Here, we use the letter '`r`' for λ; it's just a real number. Most other things here are vectors or matrices. Unfortunately, when, $r = \sqrt{|c|/|a|}$, this function typically blows up. And more interestingly, the roots we seek are quite near to this point, with one solution on the left and the other solution on the right of it. So, maybe it's not so bad that the function blows up here, after all. A simple plot clues us in immediately where we should begin to look. A huge spike marks the spot like a planted flag near a golf hole. But, that spike also affects the autoscaling of the plot function making the interesting parts of the curve difficult to see. We don't really need to see much of the spike, only get a sense of where it is. So, to enable reasonable plots to come out right, we include a modified version of, `psolq(r)`, called, `psolqcap(r)`, that simply "cap's" the function out at, $f = +1$, so we keep the range of the function within reasonable limits; $f \in [-1, +1]$. Here is the cap function. You can see what it does. This is only used for plotting.

```
function up=psolqcap(r)
global a b c;
p = psolqd(r);
pp = trace(p*p')/2;
up = pp - 1; if (up > 1) up = 1;
end
```

$$(2 + i - k)q^2 + (-3 + j + k)q + (1 - j + k) = 0 \tag{1.81}$$

So, now that we've set up our quadratic equation coefficients, and entered something like the equation above, into the global environment, we need to take a look at the plot to see where the roots are. Find the spike point, $r = \sqrt{|c|/|a|}$:

```
>> r = sqrt(sqrt(trace(c*c')/trace(a*a')))

r =

0.8409
```

That's just a little trick. The `trace(c*c')` actually computes, $2|c|^2$, but the 2's cancel in the ratio, so we get what we seek. Our '`blowhole`' is at, `r = 0.8409`. That tells us to get a nice plot we should use a wide enough range to include the two points, `r = ± 0.8409`. So, let's pick the range, `[-5,5]`. Now we set up a range variable, `rr`, find out how many data points in the range, with `size(rr)`, then call the cap function, `psolqcap()`, in a for loop to grab some data points to plot;

```
>> rr = -5:.01:5;
>> size(rr)
```

```
ans =

1 1001
>> for i = 1:1001
yy(i)=psolqcap(rr(i));
end
>>plot(rr,yy);
>>hold on
>>plot([-5 5],[0 0],'k-');
>>grid on
>>grid minor
```

We plot our curve, along with a horizontal line on the axis to see where the curve cuts the axis more clearly. Then turn on or off the grid, with, `grid on` and `grid off`, and use the minor grid if necessary, to get a better estimate of where the zero crossings are.

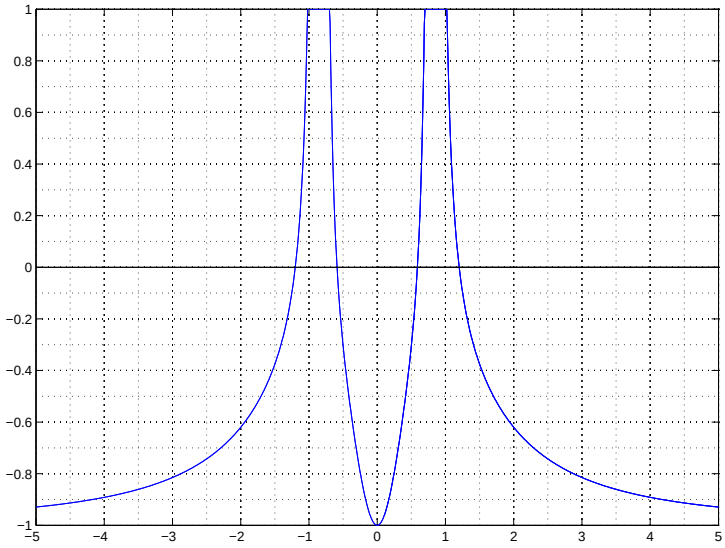


FIG. 1: Plot of psolqcap(r) , $f(\lambda) = p(\lambda)p(\lambda)^* - 1$.

Inspecting the plot, we determine the first zero is somewhere in the range, `[0.5,0.7]`, and the second zero is in the range, `[1.0,1.5]`. So, we ask MATLAB to find the exact points for us, with the `fzero()` function call. This time we use `psolq(r)`, so that MATLAB can tell us if there's a problem finding the root, such as we picking the wrong range etc...we don't need to cap things for this part.

```
>> r1 = fzero(@psolq,[0.5,0.7])
r1 =

0.5886
>> r2 = fzero(@psolq,[1.0,1.5])
r2 =

1.2013
>>
```

And we've found the two values for the magnitude, $\lambda_1 = 0.5886$, and $\lambda_2 = 1.2013$. Next up is the function, `psolqd(r)`, which takes our $\lambda's$ and computes the `direction unit` corresponding to each magnitude entered in as '`r`'.

The beauty of MATLAB here, is that we can enter the formulas exactly as we presented them in this paper itself, with almost no changes necessary. We only have to remember that the marks, `*` and `'`, are used there in code to mean, '`multiply`' and '`conjugate`', whereas in our text these marks indicate, '*conjugate*' and '*hand transform*', respectively. Now, `psolqd(r)`, returns a 2×2 complex matrix. It's really intended to be used within other functions. But, we can usefully inspect the result it provides, by calling it ourself at the command line, assigning it's output to a variable, `p = psolqd(r)`, and then checking that it's indeed a unit quaternion, by typing, `p*p'`, at the command prompt to verify we then get the unit matrix, indicating, $|p|^2 = 1$. But, it's hard to read out results in this complex matrix format, so the next function, `psolqv(r)`, does the same thing for us, except it outputs the unit direction, p, in row 'vector' format. The same format we used to input, `A, B, C`.

```
>> psolqck(r1)

ans =

1.0e-15 *

0.2220 0.1110 0 0.2220

>> psolqck(r2)

ans =

1.0e-14 *

0.1332 -0.0222 -0.0888 0

>>
```

The last two functions in this collection are, `psolqck(r)` and `psolqev(Q)`, which `check` and `evaluate`, respectively. Once we found the solution, we'd like to check that it works. A call to `psolqck(r1)`, will first compute the solution, $q = \lambda \cdot p(\lambda)$, corresponding to the magnitude, `r1`, and then call `psolqev(Q)`, with the solution encoded as a '`Q`' in row vector format, to compute the expression, $aq^2 + bq + c$, and return the result as a row vector. If everthing worked out right, we expect to see 0 here, for all 4 entires in the row vector output. As seen in the results shown above, we get very close to zero; small numbers on the order of 10^{-15} appear, because of numerical precision limits. But, for all intents and purposes, these are 0. We may set up any quaternion in row vector format, `Q`, and call, `psolqev(Q)`, ourself at the command line, at any time, to evaluate the quadratic expression, `aQ^2 + bQ + c`.

```
>> q1 = r1*psolqv(r1)

q1 =

0.0873 0.1496 -0.3279 0.4571

>> q2 = r2*psolqv(r2)

q2 =

1.0793 -0.1014 -0.4773 -0.2001

>>
```

$$(2 + i - k)q^2 + (-3 + j + k)q + (1 - j + k) = 0 \tag{1.82}$$

solutions:

$$q_1 = 0.0873 + 0.1496i - 0.3279j + 0.4571k\ , \qquad \lambda_1 = 0.5886 \tag{1.83}$$

$$q_2 = 1.0793 - 0.1014i - 0.4773j - 0.2001k\ , \qquad \lambda_2 = 1.2013 \tag{1.84}$$

The above output shows the final results, $q = \lambda \cdot p(\lambda)$, of the calculation, along with the equation solved.

Now consider the *forms* of the two following quadratic equations,

$$aq^2 + bq + c = 0 \tag{1.85}$$
$$aq^2 + bq + qb + c = 0 \tag{1.86}$$

The first form is the one we just solved. The second form is more difficult, and we haven't discussed it, yet. The latter equation has two linear terms, bq and qb, that are reverse order products. If we refer to one product as the "right action" product, $\mathrm{R}(b, q) = bq$, then by comparison the other is the "left action" product, $\mathrm{L}(b, q) = qb$. The key point, is that this b parameter is on opposite sides of the unknown, q. The mixing of left and right actions within a quaternion algebraic expression makes manipulating that expression much more difficult, because of the non-commuting nature of the product. With the hand-transformation operator $'$ we can overcome the difficulty for strictly linear problems. But, as soon as we toss a quadratic term, like, aq^2, into the mix, the difficulty returns. Notwithstanding the fact that the second equation is more difficult to solve, we have, nevertheless, just solved the first equation, which gave us two solutions, $q = q_1$ and $q = q_2$, and we can now "*construct*" yet another quadratic equation form, using these solutions, that looks alot more like the second equation above!

$$(q - q_1)(q - q_2) = 0 \quad \rightarrow \quad q^2 - q_1 q - q q_2 + q_1 q_2 = 0 \tag{1.87}$$
$$(q - q_2)(q - q_1) = 0 \quad \rightarrow \quad q^2 - q_2 q - q q_1 + q_2 q_1 = 0 \tag{1.88}$$

By this '*re-construction*' of the quadratic expression, using the two solutions, we've transformed the original quatratic, with one-sided action in the linear term, into one containing a mix of left and right actions there.

This suggests there may be a way to transform difficult quadratic equations, that mix left with right, into one-sided forms that we can solve. But, it is not clear how to do this. How can we reverse the transformation, starting with a mixed expression, apply some operations, like scaling, translation, rotation, inversion, or other, singly or in combination, to convert a mixed expression into a one-sided one? This is an open question.

Let's continue with our original equation. We required, $b \neq 0$, to arrive at this solution. We've seen that when, $b = 0$, the quadratic becomes, $aq^2 + c = 0$, which generally has two solutions with equal magnitudes. The exception is whenever the a and c parameters are such that, $a^*c \in \mathbb{R},\ a^*c > 0$, in which case, $q^2 = -1 \cdot \lambda^2$, which has an infinity of solutions. Apart from this one special exception, the equation always has two distinct roots, with same magnitude, but opposite directions. The degenerate magnitude in this case is always, $\lambda = \sqrt{|c|/|a|}$. But, this is the same critical point, in our unit direction solution, $p(\lambda)$, where our formula blows up!

We recall, our discussion from the complex number example, that when we have *same magnitude* with *different directions*, the two-step method requires special treatment. We needed '`a third step`' to find the final solution. We anticipate, then, that in this quaternion example, we'd have the same types of issues to deal with. The function that maps magnitude to direction, $\lambda \rightarrow p$, still can't give us two distinct direction units from one input magnitude. But, here we have another problem. Because of the non-commuting nature of the quaternions this time, it is a bit more challenging to take two candidate solutions and reconstruct the form of the original quadratic expression.

In complex numbers,

$$(z - z_1)(z - z_2) \;=\; z^2 - (z_1 + z_2)z + z_1 z_2 \;=\; az^2 + bz + c, \tag{1.89}$$

but, in quaternions,

$$(q - q_1)(q - q_2) \;=\; q^2 - q_1 q - q q_2 + q_1 q_2 \;\neq\; aq^2 + bq + c \quad ? *** \tag{1.90}$$
$$(q - q_2)(q - q_1) \;=\; q^2 - q_2 q - q q_1 + q_2 q_1 \;\neq\; aq^2 + bq + c \quad ? *** \tag{1.91}$$

So, let us see how we would do this in the quaternion example. Suppose, q_1 and q_2, are the two known solutions, of the quadratic equation, $aq^2 + bq + c = 0$. Instead of forming the "*reconstructed equations*" shown above, we consider the two "*direct equations*" implied by the existence of these roots.

$$aq_1^2 + bq_1 + c = 0 \tag{1.92}$$
$$aq_2^2 + bq_2 + c = 0 \tag{1.93}$$

Then, we work out the implied relationships, between the formal parameters, (a, b, c), and the roots, q_1, q_2.

$$a(q_1^2 - q_2^2) + b(q_1 - q_2) = 0 \tag{1.94}$$
$$b(q_1 - q_2) = -a(q_1^2 - q_2^2) \tag{1.95}$$
$$b = -a(q_1^2 - q_2^2)/(q_1 - q_2) = -a(q_1^2 - q_2^2)(q_1^* - q_2^*)/|q_1 - q_2|^2 \tag{1.96}$$

$$aq_1 + b + c/q_1 = 0 \tag{1.97}$$
$$aq_2 + b + c/q_2 = 0 \tag{1.98}$$
$$a(q_1 - q_2) + c(1/q_1 - 1/q_2) = 0 \tag{1.99}$$
$$c(1/q_1)(q_2 - q_1)(1/q_2) = -a(q_1 - q_2) \tag{1.100}$$
$$c = -a(q_1 - q_2)q_2[1/(q_2 - q_1)]q_1 \tag{1.101}$$
$$c = -a(q_1 - q_2)q_2(q_2^* - q_1^*)q_1/|q_2 - q_1|^2 \tag{1.102}$$
$$c = a(q_1 - q_2)q_2(q_1^* - q_2^*)q_1/|q_1 - q_2|^2 \tag{1.103}$$

Hence, we can write the following:

$$\underline{q^2 + a^{-1}bq + a^{-1}c = 0} \tag{1.104}$$

$$a^{-1}b = -(q_1^2 - q_2^2)(q_1^* - q_2^*)/|q_1 - q_2|^2 \tag{1.105}$$
$$a^{-1}c = (q_1 - q_2)q_2(q_1^* - q_2^*)q_1/|q_1 - q_2|^2 \tag{1.106}$$

Our simple *sum of the roots*, $b = -(z_1 + z_2)$, and *product of the roots*, $c = z_1 z_2$, in complex variables, become these rather more complicated formulas in quaternions, as shown above; the non-commuting nature of the algebra preventing further simplification. In non-abelian algebra, there's no factoring of the *difference of squares*, $(q_1^2 - q_2^2) \neq (q_1 + q_2)(q_1 - q_2)$, so we can't reduce the term, $a^{-1}b$, to the usual, $-(q_1 + q_2)$. But, note that the formula is, nevertheless, still symmetric in the exchange of the roots, $f(q_1, q_2) = f(q_2, q_1)$, so we can swap places for these roots and obtain the identical result. While this is obvious for the $a^{-1}b$ term, by simple inspection, it is not so clear for the $a^{-1}c$ term. But here again we can also do this exchange, yielding the alternate forms;

$$\begin{aligned} a^{-1}c &= (q_1 - q_2) \cdot q_2 \cdot (q_1^* - q_2^*) \cdot q_1/|q_1 - q_2|^2 && (1.107)\\ &= (q_2 - q_1) \cdot q_1 \cdot (q_2^* - q_1^*) \cdot q_2/|q_2 - q_1|^2 && (1.108)\\ &= (q_1 - q_2) \cdot q_1 \cdot (q_1^* - q_2^*) \cdot q_2/|q_1 - q_2|^2 && (1.109) \end{aligned}$$

The usual *product of roots*, which would simply be, $q_1 q_2$ or $q_2 q_1$, having now the two roots separated by an intervening factor, and being rather more complicated, in that we can even swap just *some* locations for, (q_1, q_2), while leaving other locations fixed! This is not at all obvious from simple inspection, and requires familiarity with these special non-abelian algrebraic expression forms; although they may be easily worked out, this still has to be done, at least once, to be convinced all these things are really the same. Thus, the effect of the extra factors is to inject some abelian character back into the otherwise non-abelian expression. Normally, $q_1 q_2 \neq q_2 q_1$, so we can't simply swap these roots in quaternions. Yet, by symmetry arguments the quadratic equation, $aq^2 + bq + c = 0$, must have an expression for the constant, c, that both "*reduces to the complex variable result*" in cases when the parameters commute, and still be "*symmetric under the exchange of the two roots*" even in non-abelian cases. Hence, by interrupting the product with these special factors, the *non-commuting*, q_1 and q_2, effectively become *commuting* again, satisfying the algebriac symmetry requirements of *interchangeable roots*!

When the roots are equal, $q_1 = q_2$, these formulas produce the indeterminate result, $0/0$. Apart from that, we can convert the formal parameters (a, b, c) into the roots, (q_1, q_2), and visa versa.

$$a, b, c \to q_1, q_2 \quad \checkmark \tag{1.110}$$
$$a, b, c \leftarrow q_1, q_2 \quad \checkmark \;\; q_1 \neq q_2 \;,\; \forall a \neq 0 \tag{1.111}$$

Only that, the reverse conversion doesn't determine a specific value for the lead parameter, a, since this can now be arbitrary, i.e. "any" a, provided it's non-vanishing, will match conditions. Next, we look at quaternion case studies.

`Case 1 - Same magnitude, different directions:`
$q_1 = \lambda \cdot u_1, q_2 = \lambda \cdot u_2, \lambda \in \mathbb{R}, \lambda > 0,\ u_1, u_2 \in \mathbb{H}_R,\ |u_1| = 1, |u_2| = 1, u_1 \neq u_2.$

We know that when, $|b| >> 0$, there are two distinct solutions, $q_1 = \lambda_1 \cdot u_1$, and, $q_2 = \lambda_2 \cdot u_2$, where, at least the two magnitudes are different, $\lambda_1 \neq \lambda_2$. In fact, we know the critical point for the spike is at, $\lambda_0 = \sqrt{|c|/|a|}$, and, therefore that, $0 < \lambda_1 < \lambda_0 < \lambda_2 < +\infty$, where, we've elected to order our solutions, so that, $\lambda_2 > \lambda_1$, by convention. We also know, that when, $b = 0$, the two magnitudes are the same, and both hit the critical point, $\lambda_1 = \lambda_0 = \lambda_2$. So, we conclude that as $|b|$ becomes smaller and smaller, approaching 0, the two solutions must move in towards each other, to converge at the same point.

$$|b| \to 0, \qquad \lambda_1 \ \to \ \lambda_0 \ \leftarrow \ \lambda_2 \ , \qquad\qquad \text{where, } \lambda_0 = \sqrt{|c|/|a|} \tag{1.112}$$

At the exact point where, $b = 0$, the solutions are of the quadratic, $aq^2 + c = 0$, which tells us, that the *unit directions*, u_1 and u_2, are precisely opposite, $u_1 = -u_2$. So, while, $|b| \to 0$, the units, u_1 and u_2, must be *rotating* relative to each other, in such fashion that, when b arrives at 0, the directions are in exact opposition. The size of the $|b|$ parameter then, determines the "width" of the spike curve. The direction of this parameter, $b/|b|$, determines the speed at which the solution directions rotate to finally converge in their final state in perfect opposition to each other; and the ratio of the two remaining parameters determines the distance, $\lambda_0 = \sqrt{|c|/|a|}$, of the spike from the origin, and the general location to start looking for the two solutions. Let us see what we learn from just looking at the plots of the '*provisional*' unit direction function, $f(\lambda) = p(\lambda)p(\lambda)^* - 1$. Consider the modified quadratic equation,

$$aq^2 + \beta \cdot bq + c = 0 \ , \qquad a, b, c, q \in \mathbb{H}_R,\ b \neq 0,\ \ \beta \in \mathbb{R} \tag{1.113}$$

We use the same numerical values for the three, a, b, c, parameters, as before, but now modify the, b, parameter by scaling its magnitude with, β. Then, $\beta = 1$, is our previous curve, and we look at, $\beta = 2$, and, $\beta = 1/4$.

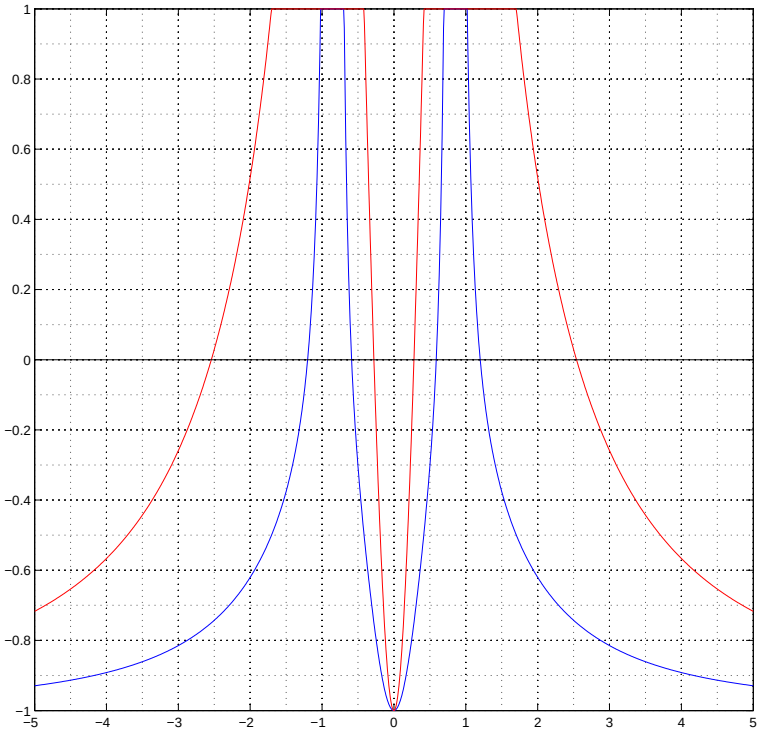


FIG. 2: Plot of psolqcap(r) , $f(\lambda) = p(\lambda)p(\lambda)^* - 1$; $aq^2 + \beta \cdot bq + c = 0$, $\beta = 1$ (blue), $\beta = 2$ (red)

.

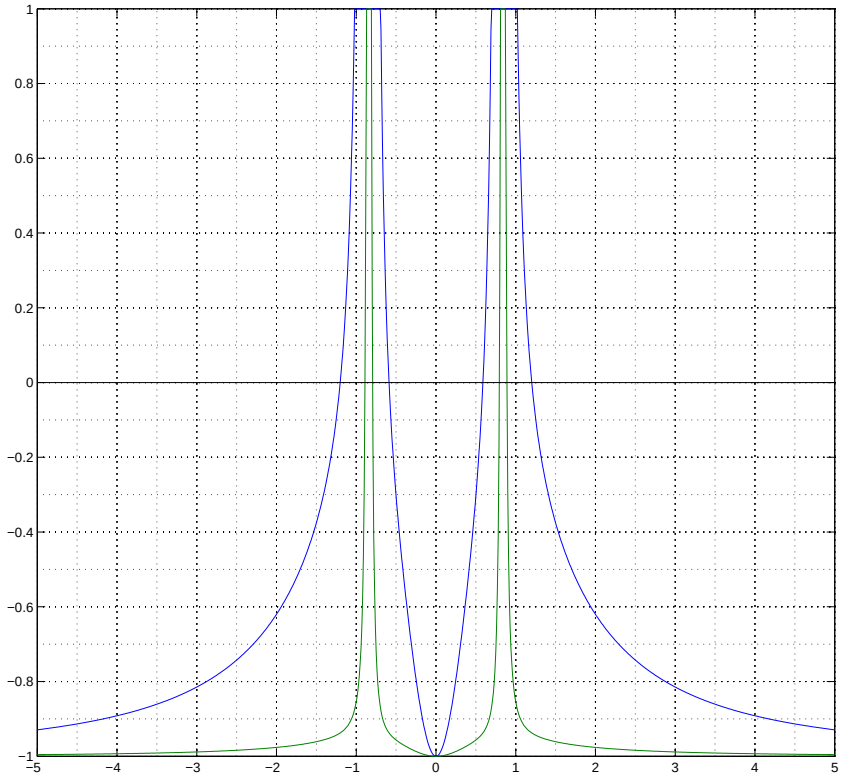


FIG. 3: Plot of psolqcap(r) , $f(\lambda) = p(\lambda)p(\lambda)^* - 1$; $aq^2 + \beta \cdot bq + c = 0$, $\beta = 1$ (blue), $\beta = 0.25$ (green)

What we observe, is that the two particular solutions here retain their distinct magnitudes, λ_1 and λ_2, all the way down, as, $|b| \to 0$, only becoming equal at the final point, $b = 0$, itself. This behavioir, of retaining the distinct separation between the magnitudes all the way down, depends on the particular choice of the known quaternions, a, b, c. As will be seen in the complex subplane example case, when the quaternions, a, b, c, fall in the same complex subplane, $a, b, c \in \mathbb{C} \subset \mathbb{H}_R$, the situation changes dramatically. The plot becomes an ordinary smooth top curve, which then begins to collapse, and the peak continues to fall, until, $|p(\lambda)| < 1$, $\forall\lambda$, so the two magnitudes merge, $\lambda_1 = \lambda_2 = \lambda$, long before the $|b|$ parameter reaches 0.

At the point where the peak is at the axis of the function, $f(\lambda) = p(\lambda)p(\lambda)^* - 1$, then, $|p(\lambda)| = 1$, and the two magnitudes become the same, $\lambda_1 = \lambda_2 = \lambda_0 = \lambda$, where, $\lambda_0 = \sqrt{|c|/|a|}$, and the unit directions also become the same, $u_1 = u_2 = p$, so that the two solutions are completely degenerate, $q_1 = q_2 = \lambda p$, having *both* their magnitude, λ, and direction, p, being identical.

From here on down, as, $|b| \to 0$ continues, the magnitudes continue to be aligned, $\lambda_1 = \lambda_2 = \lambda_0 = \lambda$, and remain that way in lock step, until, $|b| = 0$. But, the '*provisional*' unit direction function, $f(p)$, no longer has any place where it cuts or touches the axis, so, $|p| < 1$, and it's peak becomes, instead, a measure of the *average* of the two actual direction units, u_1 and u_2, i.e., $p_{max|p|} = (u_1 + u_2)/2$. While, the true units themselves, u_1 and, u_2, having started out being equal at the axis where last, $|p| = 1$, now begin to rotate away from each other becoming distinct. The degeneracy is then reserved to the magnitudes, while the complete solutions again exhibit two distinct roots, with same magnitude, but different directions. When the b parameter finally arrives at the zero point, $|b| = 0$, the two true units become exact opposites, $u_1 = -u_2$, and the problem falls out of the scope of our *two-hand two step* method. We then have to turn to the alternative *one-step* method to find the solutions. During the transition region, from the time the magnitudes first become equal, to the final state where, $|b| = 0$, the quaternion quadratic equations with these b parameters, require '`a third step`,' to find all roots. We refer to this as finding *the magic factors*.

`Finding Magic Factors`

In the complex number example, we found that we were able to recover the true direction units, u_1 and u_2, from the average, p, provided by the *two-step* method, by using a simple trick–multiplying by a pair of conjugate expressions, which we called magic factors. So, now we look for the corresponding result in quaternions.

If, $|p| < 1$, then the magnitudes are the same, but the directions differ. Find, $\max(p|(\lambda)|)$. Then, assume the provisional unit direction, $p = p(\lambda_{max})$, gives an average of the units, as it did in complex variables, then,

$$p = (u_1 + u_2)/2 \tag{1.114}$$
$$u_1 + u_2 = 2p \tag{1.115}$$
$$u_1^* + u_2^* = 2p^* \tag{1.116}$$
$$u_2 = 2p - u_1, \;\; \therefore \;\; u_2^* = 1/u_2 = 1/(2p - u_1) \tag{1.117}$$
$$u_1^* = 1/u_1 \tag{1.118}$$
$$1/u_1 + 1/(2p - u_1) = 2p^* \tag{1.119}$$
$$1 + u_1/(2p - u_1) = u_1 \cdot 2p^* \tag{1.120}$$
$$(2p - u_1) + u_1 = u_1 \cdot 2p^* \cdot (2p - u_1) \tag{1.121}$$
$$2p = u_1 \cdot 2p^* \cdot (2p - u_1) \tag{1.122}$$
$$= u_1 \cdot 2p^* \cdot 2p - u_1 \cdot 2p^* \cdot u_1 \tag{1.123}$$
$$= u_1 \cdot 4|p|^2 - u_1 \cdot 2p^* \cdot u_1 \tag{1.124}$$
$$2p = 4|p|^2 \cdot u_1 - u_1 \cdot 2p^* \cdot u_1 \tag{1.125}$$
$$u_1 \cdot 2p^* \cdot u_1 - 4|p|^2 \cdot u_1 + 2p = 0 \tag{1.126}$$
$$u_1 \cdot p^* \cdot u_1 - 2|p|^2 \cdot u_1 + p = 0 \tag{1.127}$$

Since the initial equation is completely symmetric in the interchange of the two units, u_1 and u_2, a similar quadratic is obtained for u_2, and both units are therefore the solutions to the same quadratic equation, which is,

$$up^*u - 2|p|^2u + p = 0 \tag{1.128}$$

This result is similar to the equation obtained in the complex variable example, where we had, $p^*u^2 - 2|p|^2u + p = 0$, except here, because of the non-commuting nature of the general quaternions, we have a different type of square term, up^*u, which leads to another type of quaternion quadratic equation, of form, $qaq + bq + c = 0$. This new type of equation we will deal with separately, in a more general manner, later below. Here, however, the $b = -2|p|^2 \in \mathbb{R}$, is a real valued parameter, and the equation simplifies easily, into something we can readily solve right here. We start with a change of variable, $U = p^*u$,

$$U = p^*u \tag{1.129}$$
$$p\,U = |p|^2u \tag{1.130}$$
$$u = |p|^{-2}pU \tag{1.131}$$
$$\underline{up^*u - 2|p|^2u + p = 0} \tag{1.132}$$

$$\therefore \quad |p|^{-2}pUU - 2pU + p = 0 \tag{1.133}$$
$$|p|^{-2}U^2 - 2U + 1 = 0 \tag{1.134}$$
$$U^2 - 2|p|^2U + |p|^2 = 0 \tag{1.135}$$
$$(U - |p|^2)^2 - |p|^4 + |p|^2 = 0 \tag{1.136}$$
$$U = |p|^2 \pm \sqrt{|p|^4 - |p|^2} \tag{1.137}$$
$$U = |p|^2(1 \pm \sqrt{-1}\sqrt{|p|^{-2} - 1}) \;\; , \qquad |p| < 1, \;\; \therefore \;\; |p|^{-2} - 1 > 0 \tag{1.138}$$
$$U = |p|^2(1 \pm n \cdot \sqrt{|p|^{-2} - 1}) \;\; , \qquad n = n_1 i + n_2 j + n_3 k, \; n_1, n_2, n_3 \in \mathbb{R}, \; n_1^2 + n_2^2 + n_3^2 = 1 \tag{1.139}$$

$$\therefore \quad u = p \cdot (1 \pm n \cdot \sqrt{|p|^{-2} - 1}) \tag{1.140}$$

An infinity of solutions, $u = p \cdot (1 \pm n\sqrt{|p|^{-2} - 1})$, where, $n = (n_1 i + n_2 j + n_3 k)$, $n_1, n_2, n_3 \in \mathbb{R}$, solves the quadratic equation, $up^*u - 2|p|^2u + p = 0$. We over specify them with the $\pm$ signs, but this is just for convenience. There really

is just one infinite set, when it's infinity, but n is a bit ambiguous here. We still have to determine what values of n will satisfy the original quadratic, given particular values of the three, a, b, c.

`Onplane Magic Factors`

Now, there are two special solutions, that correspond to the complex number case, when, a, b, c are in the same complex subplane, $\mathbb{C} \subset \mathbb{H}_R$. Then, the arbitrary unit imaginary, n, is restricted to the same complex plane, and is along the same direction as the vector part of p, i.e., $n = \pm(p - p^*)/|p - p^*|$. This is because a finite number of $+, -, \times, \backslash, /$, operations are performed on the three, a, b, c, to produce, p, q, so the latter must be in the same complex space as the former. We can't add, subtract, multiply, or divide two complex numbers and obtain a number outside the complex plane, and there are no square roots in the formulas for, $p(\lambda)$, or, $q(\lambda, p) = \lambda \cdot p(\lambda)$.

When, $a, b, c \in \mathbb{C} \subset \mathbb{H}_R$, then, $p = p(\lambda; a, b, c) \in \mathbb{C} \subset \mathbb{H}_R$, and, $q = \lambda p \in \mathbb{C} \subset \mathbb{H}_R$.

$$q_1 = \lambda \cdot u_1 = \lambda \cdot p \cdot \left(1 + \frac{(p - p^*)}{|p - p^*|} \sqrt{|p|^{-2} - 1} \right) \tag{1.141}$$

$$\tag{1.142}$$

$$q_2 = \lambda \cdot u_2 = \lambda \cdot p \cdot \left(1 - \frac{(p - p^*)}{|p - p^*|} \sqrt{|p|^{-2} - 1} \right) \tag{1.143}$$

Actually, there is a subtle issue here. We did just introduce a square root operation, in calculating, u, from, p. It is the square root that can spring us out of a lower dimensional space, into a higher dimension. So, despite the fact that, a, b, c, p, q, must all be in the same complex subplane, by the regular laws of algebra, the final, q_1 and q_2, might not be in the same plane! They could have exactly offsetting components, perpendicular to the given plane, so that, their average is within the plane, $q = (q_1 + q_2)/2 \in \mathbb{C},\ q_1, q_2 \notin \mathbb{C}$, and the average of their unit directions, are again the same, $p = (u_1 + u_2)/2 \in \mathbb{C},\ u_1, u_2 \notin \mathbb{C}$. We are, after all, sitting in the *full right hand* quaternion space, $\mathbb{H}_R$, despite the complex plane restriction on the input parameters, and we just encountered, $\sqrt{-1}\sqrt{|p|^{-2} - 1}$.

There is a certain ambiguity whenever the square root of a negative quantity appears in a calculation. The $\sqrt{-1}$ could be interpreted variously. We'll see later, when we come to the reverse quadratic equation, $q^2 a + qb + c = 0$, how Hamilton treated the $\sqrt{-1}$ in an example he qave, electing to introduce *biquaternions*, to resolve the question of representing a new imaginary that *commutes* with, i, j, k, when he could have just used the left hand quaternions, i', j', k', alternatively, had he not forgotton about them! So, the issue is partly one of establishing a convention, and making a choice, each time the $\sqrt{-1}$ is encountered while manipulating expressions. We decide on the space we're working in, and elect to restrict results to that space. Currently, we're working in $\mathbb{H}_R$, so it is completely possible and acceptable to interpret the root to produce results not '`onplane`.' However, for the moment, we'll assume that they are. We deal with the full '`offplane`' case later.

We should note that, because the complex variable algebra is a subset of quaternion algebra, but the complex variable algebra is itself complete, by the *Fundamental Theorem of Algebra*, there must be two solutions to this quadratic equation within the complex subplane. There could be more solutions, outside the plane. But, at least these two roots have to exist onplane, and they are given by the formulas above. In those formulas, we could use any *non-scalar* parameter from, a, b, c, p, q, to construct the plane's imaginary unit, n. It doesn't have to be, p, as we have shown, since the vector parts of all these quaternion parameters are linearly dependent. However, the special "fix up" factor, $\sqrt{|p|^{-2} - 1}$, appearing in the numerator, is always a function of the provisional unit, p, with, $|p| < 1$.

The other point to make here, is that although it is technically possible to have roots offplane, most of the time *this* quadratic equation only has two roots; and when it has more, it has an infinity of roots. Therefore, apart from the exceptional case where infinity is the count, the two roots must be the very two onplane solutions given above. When the count is infinity, these are again the two roots out of that infinity, that appear on the plane. So, even though we'll construct a more sophisticated formula for all offplane and onplane roots, in the next section below, this simplified onplane solution is all we typically need for the subplane cases. Although Hamilton does give 'six' roots for an example of a somewhat similar quadratic equation, as we'll discuss later, 4 of these extra roots are biquaternions. In the quaternion over real valued coefficients, there are just two roots or infinity.

`A Practical Example:` Let us now consider an example to explore these ideas graphically. We're going to artificially construct a quaternion quadratic equation with equal magnitude solutions. First we consider a particular way of doing this construction for equations of the form, $aq^2 + bq + c = 0$, where we deliberately restrict things to

a complex subplane, and work out the details similar to the complex number example. So, we first construct equal magnitudes for a complex variable quadratic equation. Then, we put that complex plane into the quaternion space, and extend the unknown variable from $\mathbb{C}$ to $\mathbb{H}_R$.

Using the squares, $4^2 + 3^2 = 5^2$, $12^2 + 5^2 = 13^2$, and, $2^2 + 3^2 + 6^2 = 7^2$, we make up a convenient pair of unit quaternions, $u_1 = (4 - 3n)/5$, and, $u_2 = (12 + 5n)/13$, with pure unit imaginary, $n = (3i - 2j + 6k)/7$, where we know all are perfectly normalized to unity, and then set the same magnitude to be, $\lambda_1 = \lambda_2 = 5$, to invent two candidate solutions, first in complex numbers, $z_1 = \lambda_1 u_1 = 5 \cdot (4 - 3i)/5$, and, $z_2 = \lambda_2 u_2 = 5 \cdot (12 + 5i)/13$, then, convert over to quaternions, by replacing the unit, $i \to n$, for our test case. In the complex plane, we construct our parameters, $a = 1,\ b = -(z_1 + z_2)$, $c = z_1 z_2$, to obtain our three parameters, a, b, c, then construct the complex quadratic equation, $az^2 + bz + c = 0$, put it in the quaternion space on a subplane domain, and then extend the domain of the unknown, $z \to q,\ z \in \mathbb{C},\ q \in \mathbb{H}_R$, to obtain, $aq^2 + bq + c = 0$. With this procedure, we circumvent the issues in converting, a mixed action non-abelian expression like, $(q - q_1)(q - q_2)$, into a left side only expression like, $aq^2 + bq + c$.

$$aq^2 + bq + c = 0 \tag{1.144}$$

$$a = 1 \tag{1.145}$$
$$b = (-112 + 6i - 4j + 12k)/13 \tag{1.146}$$
$$c = (2205 - 240i + 160j - 480k)/91 \tag{1.147}$$

Here is the plot for this quadratic equation, with same magnitudes, different directions. A smooth curve with peak

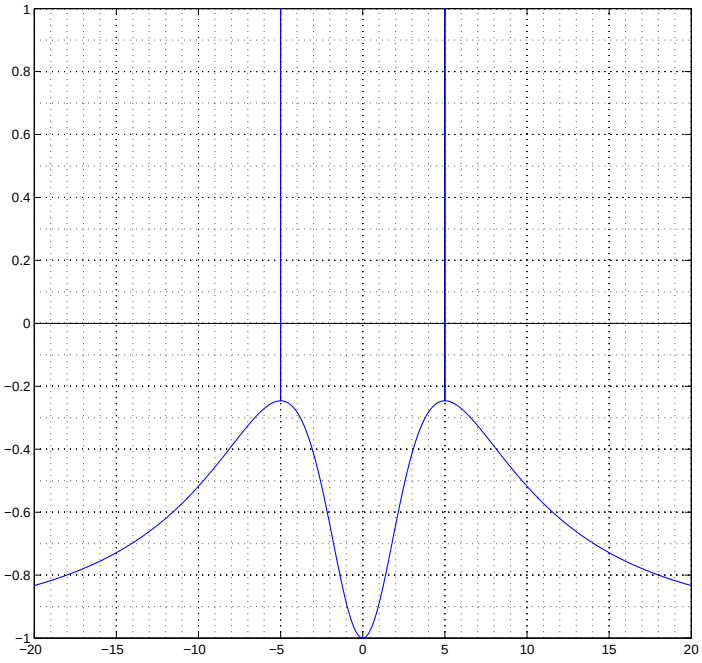


FIG. 4: Plot of psolqcap(r) , $f(\lambda) = p(\lambda)p(\lambda)^* - 1$; $aq^2 + bq + c = 0$, same magnitudes, different directions

below the axis defines the type of curve in the '`onplane`' equal magnitudes case.

The spikes seen on the hill tops are artificial. They are not really there in the theory. They result from the finite precision limits of the computer. These artifacts would be removed in implementing production code. But, here we leave them in, for educational value. Every arithmetic operation performed by the computer has some precision error. Usually, these are too small to be noticed. However, in this case, as we approach the hill top, both the numerator and denominator approach 0. In theory, they approach zero at the same rate, so the ratio should remain

finite, and there should be no spike. But, in practice, since the numerator here has very many more arithmetic calculations than the denominator, the cummulative error in the numerator grows bigger relative to the error in the denominator as we approach the critical point. So, they stop going to zero at the same rate. This happens suddenly, as soon as the precision limits are reached in the dive to 0, and causes the calculation to appear to be approaching a divide by zero state, hence the spike. The code would be modified using *L'Hôspital's Rule* to workaround the problem.

```
>> global a b c
>> A = [1,0,0,0];
>> B = [-112,6,-4,12]/13;
>> C = [2205,-240,160,-480]/91;
>> psolqabc(A,B,C)
>> r = sqrt(sqrt(trace(c*c')/trace(a*a')))

r =

5

  >> rr = -20:0.001:20;
>> size(rr)

ans =

1 40001

>> for i = 1:40001
yy(i)=psolqcap(rr(i));
end
>> plot(rr,yy)
>> hold on
>> plot([-20 20],[0 0],'k-')
>> grid on
>> grid minor
>>
```

When we compute the magnitude we get, $\lambda = r = \sqrt{|c|/|a|} = 5$. Of course, we set the magnitude to 5. And the peak of the curve is at 5, and the artificial spike is at 5. If we tried to read out the value of $p(\lambda = 5)$, at the theoretical max $|p(\lambda)|$, in MATLAB, this is what we'd get,

```
>> p = psolqv(5)

p =

Inf -Inf Inf -Inf

>> p = psolqv(5.0000000001)

p =

0.8615 -0.0462 0.0308 -0.0923
>>
```

To work around, we could modify the code. But first, we use a simple trick, of adding a small number to the 5 to read out just near to the peak of the smooth curve, avoiding the spike. We finally get to see the true value of p, at $max(|p(\lambda)|)$; the *provisional* direction unit is, $p = (0.8615 - 0.0462i + 0.0308j - 0.0923k)$. So, let's compute, $|p|^2$.

```
>> pp = p*p'

pp =

0.7538
```

This confirms our analysis, that when equal magnitudes occur, $|p|^2 < 1$. Here, it's, $|p|^2 = 0.7538$. So, we can now use this p to construct our magic factors, `m1` and `m2`. First, we need the vector part, `Vp = (p - p*)/2`, then we normalize to obtain the pure imaginary unit, `nVp = Vp/|Vp|`, then calculate the magic.

```
>> Vp = p
                                        Vp = 0.8615 -0.0462 0.0308 -0.0923
>> Vp(1) = 0
                                        Vp = 0 -0.0462 0.0308 -0.0923
>> nVp = Vp/sqrt(Vp*Vp')
                                        nVp = 0 -0.4286 0.2857 -0.8571
>> m1 = [1,0,0,0] + nVp*sqrt(1/pp - 1)
                                        m1 = 1.0000 -0.2449 0.1633 -0.4898
>> m2 = [1,0,0,0] - nVp*sqrt(1/pp - 1)
                                        m2 = 1.0000 0.2449 -0.1633 0.4898
>>
```

Notice that the complex subplane's imaginary unit, `nVp`, in the solution, p, is the same, n, that we constructed, to build the input parameters, a, b, c, with, up to a sign, $nVp = \pm n$. Checking, $n = (3i - 2j + 6k)/7$;

```
>> n = [0,3,-2,6]/7
                                        n = 0 0.4286 -0.2857 0.8571
>>
```

So, now we know the degenerate magnitude, $\lambda = r = 5$, we know the average of the unit directions, $p = (u_1 + u_2)/2$, we know the conjugate pair of magic factors, `m1` and `m2`, lets build the final solutions. [!.lookahead pg.99, for mulq()]

```
>> u1 = mulq(p,m1)
                                        u1 = 0.8000 -0.2571 0.1714 -0.5143
>> u2 = mulq(p,m2)
                                        u2 = 0.9231 0.1648 -0.1099 0.3297
>> q1 = r*u1
                                        q1 = 4.0000 -1.2857 0.8571 -2.5714
>> q2 = r*u2
                                        q2 = 4.6154 0.8242 -0.5495 1.6484
>>
```

Let's see, we started out with invented solutions, $Q_1 = 5 \cdot (4 - 3n)/5$, and, $Q_2 = 5 \cdot (12 + 5n)/13$, where, the imaginary unit was, $n = (3i - 2j + 6k)/7$. So, check,

```
>> Q1 = 5*[ [4,0,0,0] - 3*[0,3,-2,6]/7 ]/5
                                        Q1 = 4.0000 -1.2857 0.8571 -2.5714
>> Q2 = 5*[ [12,0,0,0] + 5*[0,3,-2,6]/7 ]/13
                                        Q2 = 4.6154 0.8242 -0.5495 1.6484
>>
```

Success! Well, not quite. Although things look the same, there's actually a huge error hidden in the numbers. It's just below the level of precision displayed on the screen. Let's see what happens, when we try to verify these solutions, by plugging them back into the original quadratic equation. `Verify results...`

```
>> v1 = mulq(A,mulq(q1,q1)) + mulq(B,q1) + C          ...or use:>> v1 = psolqev(q1), etc..
                                        v1 = 1.0e-04 * [ -0.2599 0.1075 -0.0717 0.2151 ]
>> v2 = mulq(A,mulq(q2,q2)) + mulq(B,q2) + C
                                        v2 = 1.0e-05 * [ 0.5396 0.0641 -0.0427 0.1282 ]
>> V1 = mulq(A,mulq(Q1,Q1)) + mulq(B,Q1) + C
                                        V1 = 1.0e-14 * [-0.3553 -0.0444 -0.1776 -0.0888 ]
>> V2 = mulq(A,mulq(Q2,Q2)) + mulq(B,Q2) + C
                                        V2 = 1.0e-14 * [-0.3553 0.0444 0 0.0888 ]
>>
```

The solution we "found," evaluates the expression, $aq^2 + bq + c$, to, 0, to the level of, 10^{-5}, but the solution that we "put in," to construct the problem, evaluates the expression to, 0, closer to the actual level of precision of the machine itself, 10^{-15}. That's 10 orders of magnitude different, 10^{10}. A ginormous error! What went wrong?

Well, we know we cheated a little. We didn't use, $\lambda = r = 5$, in our work to find p. Instead, we shifted it a bit, to avoid that spike, and used, $\lambda = r = 5.0000000001 = 5 + 10^{-10}$. Ah, ha! We put in that error to read out the value of p near the max. We thought that was a small number. But, somewhere, somehow, that, 10^{-10}, got inverted, $1/(10^{-10}) = 10^{+10}$, during the calculation,[4] and the final resulting effect was huge. But, we can't read out the p at the true λ-point, because the spike is there, messing things up. So, we have no choice but to get rid of the spike.

Later, in the "Vanishing Denominator" case below we'll give the *L'Hôspital's Rule*, updated formula, and MATLAB code to solve this problem. But, right now, we'll just use the new function presented there to check this out more thoroughly. The new function is called, `psolqvh(r)`, and it just calculates `psolqv(r)`, in *L'Hôspital's* style; hence the appended 'h'. We only use this function at the critical point itself. It's intended to "*crush the spike*," and enable us to read out the true value sitting right under that spike.

```
>> r
r =
5
>> p = psolqvh(r)
p =
0.8615 -0.0462 0.0308 -0.0923
```

Checking that, $r = 5$, we call, `psolqvh(r)`, and notice there's no apparent change in output from the last call to, `psolqv(5.0000000001)`. That's because we're now flying under the radar. We repeat all the steps shown previously, up to building the final solutions, `q1` and `q2`, with the newly updated magic factors. All outputs to screen look exactly the same as before, until we come to the last step. `Verify results...`

```
>> v1 = mulq(A,mulq(q1,q1)) + mulq(B,q1) + C
                                        v1 = 1.0e-13 * [ -0.5329 0.2265 -0.2043 0.4174 ]
>> v2 = mulq(A,mulq(q2,q2)) + mulq(B,q2) + C
                                        v2 = 1.0e-13 * [ -0.7816 -0.0488 0.0711 -0.1332 ]
>>
```

Here is where we see the huge change. We're alot closer to 0, at the level of, 10^{-14}, just a factor of 10 away from the best possible results, at, 10^{-15}. We expect some error, given the large number of arithmetic computations invloved in manipulating the many 2×2 complex number matrices, in the numerator and denominator, to arrive at this point, knowing that the finite precision of the machine must introduce some cumulative error in the process. Even though we deliberately picked special whole number pythagorean triples, to construct fault free unit quaternions, to reduce the number of precision errors as much as possible on input parameters, we can't really expect that level of precision would be maintained all the way through the many calculations to arrive at the end with the exact numbers we entered in to begin with. All things considered, things are looking good!

```
>> q1*7*5
              ans = 140.0000 -45.0000 30.0000 -90.0000
>> q2*7*13
              ans = 420.0000 75.0000 -50.0000 150.0000
>>
```

A final check. We multiply our solutions, `q1` and `q2`, by the divisors used in initial construction, to see how close we get to the original integer numbers. At the level of default screen output precision, the match is exact. We'd have to enter `>> format long` at the command prompt to see the real difference, way down in the 15th decimal place.

`Changing the graphic.` Let's consider some modified quadratic equations of the form, $aq^2 + \beta \cdot bq + c = 0$, for this special `onplane` equal magnitudes practical example; and then compare the onplane equation with our previous equation, to see the differences. We're going to scale, b, up and down, to see how the plots change.

---

[4] Actually, the matching 10^{10} numbers here is pure coincidence. Setting, $\lambda = r = 5 + 10^5$, does produce a better result, 0, to the level of, 10^{-10}, which is only, 10^5, larger than the the precision level. But, this is then the optimal place to grab the $p(\lambda)$ from. Moving farther away, doesn't improve the results. We were just too close to the spike with $5 + 10^{-10}$.

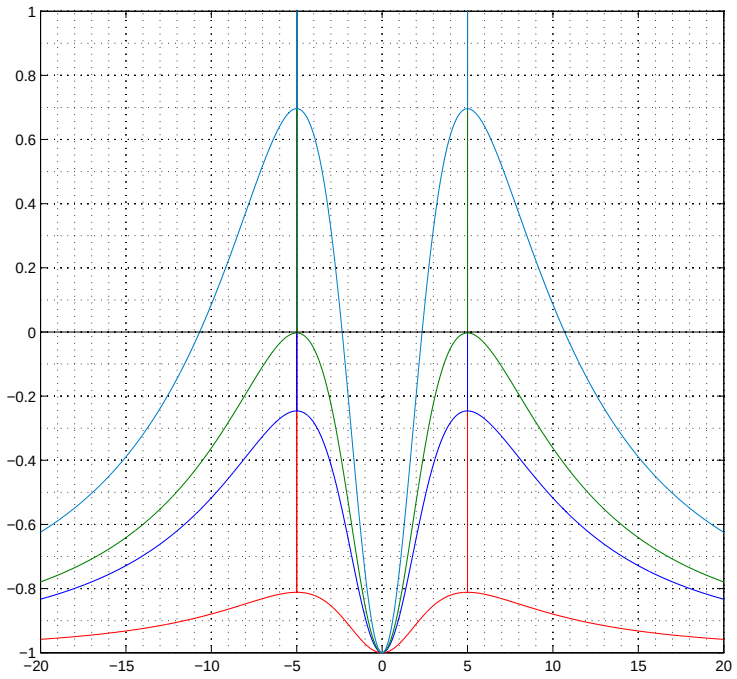


FIG. 5: Plot of psolqcap(r) , $f(\lambda) = p(\lambda)p(\lambda)^* - 1$; $aq^2 + \beta \cdot bq + c = 0$, onplane hill tops, showing variation with $\beta \cdot b$ parameter; $\beta = 0.5$ (red), $\beta = 1.0$ (blue), $\beta = 1.15$ (green), $\beta = 1.5$ (cyan).[Like blackbody radiation curves, except peaks fixed by $|c|/|a|$.]

Here we have the plots of four curves. They graphically illustrate the effect of changing the magnitude of the linear coefficient, b, in the quadratic equation. Recall, in the first quadratic example we considered, there was a huge natural spike, and as we changed the magnitude of the b parameter there, the spike just got thinner and thinner, as the two solutions moved in towards each other. The height of the spike itself, didn't move up or down. Just the width of the curve was affected. Here, in the onplane example, the hill top moves up and down, as we change the magnitude of b.

$$\begin{aligned} a_1 &= 2 + i - k \quad , & a_2 &= 1 \ , \qquad r = \sqrt{|c_1|/|a_1|} = 0.8409 \ , \\ b_1 &= -3 + j + k \ , & b_2 &= -(r \cdot (4 - 3 \cdot (3i - 2j + 6k)/7)/5 + r \cdot (12 + 5 \cdot (3i - 2j + 6k)/7)/13) \ , \\ c_1 &= 1 - j + k \quad , & c_2 &= +(r \cdot (4 - 3 \cdot (3i - 2j + 6k)/7)/5) \cdot (r \cdot (12 + 5 \cdot (3i - 2j + 6k)/7)/13) \ , \end{aligned} \tag{1.148}$$

$$a_1 q^2 + \beta_1 \cdot b_1 q + c_1 = 0 \qquad \leftarrow \text{ deep space natural spike} \tag{1.149}$$

$$a_2 q^2 + \beta_2 \cdot b_2 q + c_2 = 0 \qquad \leftarrow \text{ onplane hill top with artificial spike} \tag{1.150}$$

$$a_3 = 0.01 \cdot a_1 + 0.99 \cdot a_2 \tag{1.151}$$

$$b_3 = 0.01 \cdot 0.25 \cdot b_1 + 0.99 \cdot 1.00 \cdot b_2 \tag{1.152}$$

$$c_3 = 0.01 \cdot c_1 + 0.99 \cdot c_3 \tag{1.153}$$

$$a_3 q^2 + b_3 q + c_3 = 0 \qquad \leftarrow \text{ hybrid: near plane hill top with natural spike} \tag{1.154}$$

Now we'd like to put the two type of curves on the same graph, to compare them more closely. To do this we rescale the magnitudes of our onplane equation parameters, reducing from $\lambda = 5$, to match the first equation, so both have the same critical point, at $\lambda = r = \sqrt{|c_1|/|a_1|} = 0.8409$. We pick $\beta_1 = 0.25$, to get a nice thin natural spike in the (a_1, b_1, c_1)–equation, and keep, $\beta_2 = 1.0$, so our onplane parameters, (a_2, b_2, c_2), remains our default curve. Then we build a hybrid quadratic equation, with parameters, (a_3, b_3, c_3), combining these two, with 1% natural spike and 99% hill top, to construct a hill top curve that breaks out into a natural spike, to contrast with the hill top with artificial spike produced by the machine. The (a_3, b_3, c_3) move off the subplane, becoming `near-plane` instead.

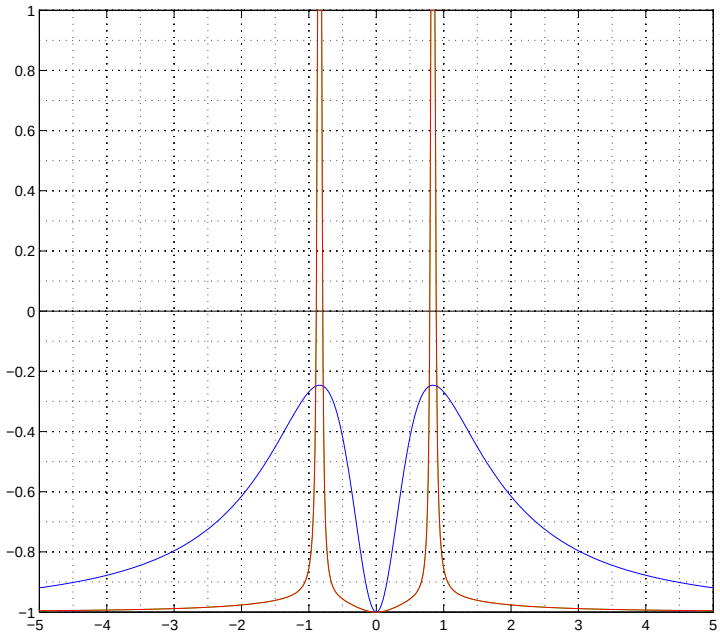


FIG. 6: Plot of psolqcap(r) , $f(\lambda) = p(\lambda)p(\lambda)^* - 1$; onplane hill top (blue) vs deep space natural spike (red)

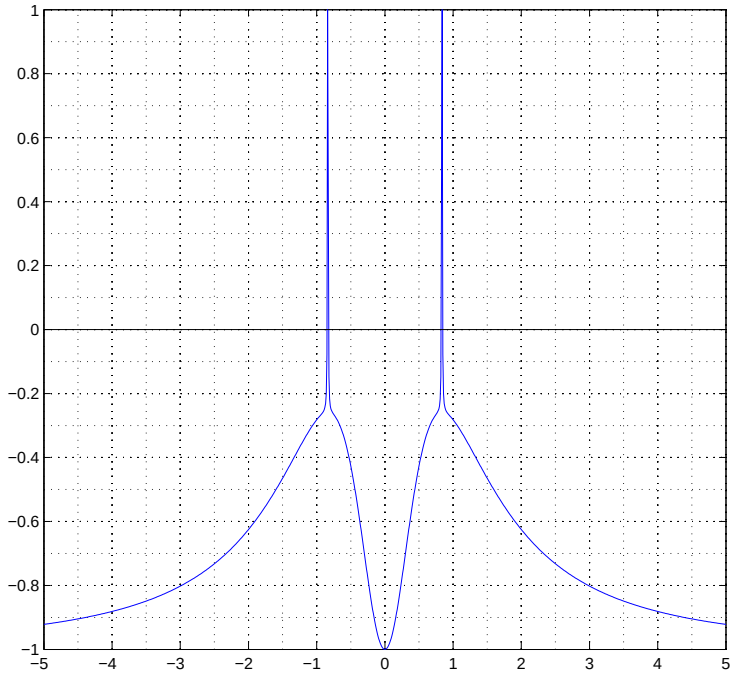


FIG. 7: Plot of psolqcap(r) , $f(\lambda) = p(\lambda)p(\lambda)^* - 1$; $a_3q^2 + b_3q + c_3 = 0$, hybrid: near-plane hill top with natural spike!

`Deep Space.` In the first quadratic equation example, the three parameters, (a_1, b_1, c_1), are all uncorrelated. They are not on a complex subplane, nor near a complex subplane, they are out there in deep space in the quaternion 4-dimensional world, off in somewhat randomly chosen directions. We'll call this the "deep space" equation. The provisional unit curve, $f(\lambda) = p(\lambda)p(\lambda)^* - 1$, is characteristically spiked, breaking though the axis, providing two distinct solutions with different magnitudes, $\lambda_1 \neq \lambda_2$, one on either side of the center of the spike. This is a natural spike, that is a proper property of the algebraic expression involved in the equation, and is a necessary part of the curve that helps to locate the true solutions. The linear parameter, b, determines the "width" of this natural spike, and as, $|b| \to 0$, the width gets smaller and smaller, while the two values, λ_1 and λ_2, remain distinct, although all while getting closer, all the way down until $|b| = 0$. At that point, they become exactly equal. But, just prior to that, they are infinitesimally close, and the spike is infinitesimally thin.

`Onplane.` In the second quadratic equation example, the three parameters, (a_2, b_2, c_2), are all correlated. They sit on a particular complex subplane within the quaternion space, and though they may have different quaternion directions within the plane, their vector parts are all aligned to some effective complex imaginary unit, n. Changing the magnitude of the linear coefficient, b, here, does a similar thing as before, in that when, $|b| >> 0$, decreasing $|b|$, decreases the separation between the two magnitudes, λ_1 and λ_2, so the "width" of the curve at the axis is affected similarly to the deep space natural spike. But, here there's also a hill top that falls with decreasing $|b|$, so that long before $|b| = 0$, the two solution magnitudes have merged into one, $\lambda_1 = \lambda_2$, and they remain that way from there on, all the way down until $|b| = 0$.

`Near-Plane.` In the hybrid quadratic equation example, the three parameters, (a_3, b_3, c_3), are almost correlated. They are very near to a complex subplane within the quaternion space, and would be onplane, were it not for the fact that they were deliberately shifted out of the plane a small amount in the direction of the deep space parameters, (a_1, b_1, c_1), to construct this composite equation. We'll call this the "near-plane" equation. The curve here happens to look like the onplane hill top, because it is very near to being onplane, but at the same time the hill top breaks out into a genuine bona fide natural spike, sitting right where the artifical spike would normally be, causing a bit of confusion, when interpreting the curves. We now have to decide when it is right to crush the spike with *L'Hôspital's* hammer.

Let's see how we would approach this problem. There are two possibilities, when we see such a curve in our plots. `(1) Spike is an artifact`, we remove it and read out the value under the spike, to find the *degenerate* solutions. `(2) Spike is genuine`, we find where it cuts the axis, and read out those values to find the two *distinct* solutions.

```
>> global a b c
>> A1 = [2,1,0,-1];
>> B1 = [-3,0,1,1];
>> C1 = [1,0,-1,1];
>> psolqabc(A1,B1,C1)
>> r = sqrt(sqrt(trace(c*c')/trace(a*a')))

r =

0.8409

>> A2 = [1,0,0,0];
>> B2 = -r*(([4,0,0,0] - 3*[0,3,-2,6]/7)/5 + ([12,0,0,0] + 5*[0,3,-2,6]/7)/13);
>> C2 = mulq(r*([4,0,0,0] - 3*[0,3,-2,6]/7)/5, r*([12,0,0,0] + 5*[0,3,-2,6]/7)/13);
>>
>> A3 = 0.01*A1 + 0.99*A2;
>> B3 = 0.01*0.25*B1 + 0.99*1.00*B2;
>> C3 = 0.01*C1 + 0.99*C2;
>>
>> psolqabc(A3,B3,C3)
>> rr = -5:.01:5;
>> size(rr)

ans =
```

```
1 1001

>> for i = 1:1001
yy(i)=psolqcap(rr(i));
end
>> plot(rr,yy)
>> hold on
>> plot([-5 5],[0 0],'k-')
>> grid on
>> grid minor
>> r = sqrt(sqrt(trace(c*c')/trace(a*a')))

r =

0.8367

>>
```

So, we've set up our curve. Although we started out with, $\lambda = r = 0.8409$, our combination into hybrid shifted the critical point a bit over to, $\lambda = r = 0.8367$. This is where our spike is now. Let's check to see.

```
>> p = psolqv(r)

p =

-Inf -Inf Inf -Inf

>>
```

Crush the spike with *L'Hôspital's* hammer, and read out the value of $p(\lambda)$ under it. Then, build the solutions.

```
>> p = psolqvh(r)
                                                    p = 0.8497 -0.0449 0.0512 -0.1238
>> pp = p*p'
                                                    pp = 0.7419
>> Vp = p
                                                    Vp = 0.8497 -0.0449 0.0512 -0.1238
>> Vp(1) = 0
                                                    Vp = 0 -0.0449 0.0512 -0.1238
>> nVp = Vp/sqrt(Vp*Vp')
                                                    nVp = 0 -0.3178 0.3621 -0.8763
>> m1 = [1,0,0,0] + nVp*sqrt(1/pp - 1)
                                                    m1 = 1.0000 -0.1874 0.2136 -0.5169
>> m2 = [1,0,0,0] - nVp*sqrt(1/pp - 1)
                                                    m2 = 1.0000 0.1874 -0.2136 0.5169
>> u1 = mulq(p,m1)
                                                    u1 = 0.7663 -0.2042 0.2326 -0.5630
>> u2 = mulq(p,m2)
                                                    u2 = 0.9330 0.1144 -0.1303 0.3154
>> q1 = r*u1
                                                    q1 = 0.6412 -0.1708 0.1946 -0.4710
>> q2 = r*u2
                                                    q2 = 0.7806 0.0957 -0.1090 0.2639
>> v1 = mulq(A3,mulq(q1,q1)) + mulq(B3,q1) + C3
                                                    v1 = -0.0203 -0.0032 -0.0028 0.0395
>> v2 = mulq(A3,mulq(q2,q2)) + mulq(B3,q2) + C3
                                                    v2 = 0.0355 0.0067 -0.0245 0.0092
>>
```

Well, our two solutions, q_1 and, q_2, evaluate the quadratic expression, $aq^2 + bq + c$, to, 0, to the level of 10^{-2}. Not looking so good. Let's try the alternative hypothesis. Spike is genuine, find the axis points. Now, this is hard to read

off the plot, since the spike is so thin. But, we know where it is exactly, already. So, we can just pick a range just to the left of the known critical point, and another range just to the right, and ask MATLAB to figure out the rest.

```
>> r1 = fzero(@psolq,[r-.1,r-10^-5])
                                                    r1 = 0.8257
>> r2 = fzero(@psolq,[r+10^-5,r+.1])
                                                    r2 = 0.8478
>> u1 = psolqv(r1)
                                                    u1 = 0.9117 0.2356 -0.0801 0.3269
>> u2 = psolqv(r2)
                                                    u2 = 0.7966 -0.2849 0.1630 -0.5076
>> q1 = r1*u1
                                                    q1 = 0.7528 0.1945 -0.0661 0.2699
>> q2 = r2*u2
                                                    q2 = 0.6754 -0.2416 0.1382 -0.4304
>> v1 = mulq(A3,mulq(q1,q1)) + mulq(B3,q1) + C3
                                                    v1 = 1.0e-14 *[ 0.5440 0.2401 -0.0951 0.3247 ]
>> v2 = mulq(A3,mulq(q2,q2)) + mulq(B3,q2) + C3
                                                    v2 = 1.0e-13 *[ -0.1288 0.0382 -0.0206 0.0719 ]
>>
```

Now we have solutions, q_1 and, q_2, that evaluate the expression, $aq^2 + bq + c$, to, 0, back down closer to the level of 10^{-14}. This is where we'd expect the true solutions to be. Hence, our conclusion? We must reject the hypothesis that spike is artifact, and accept it as genuine.

```
Hybrid #2:  A closer near-plane
```

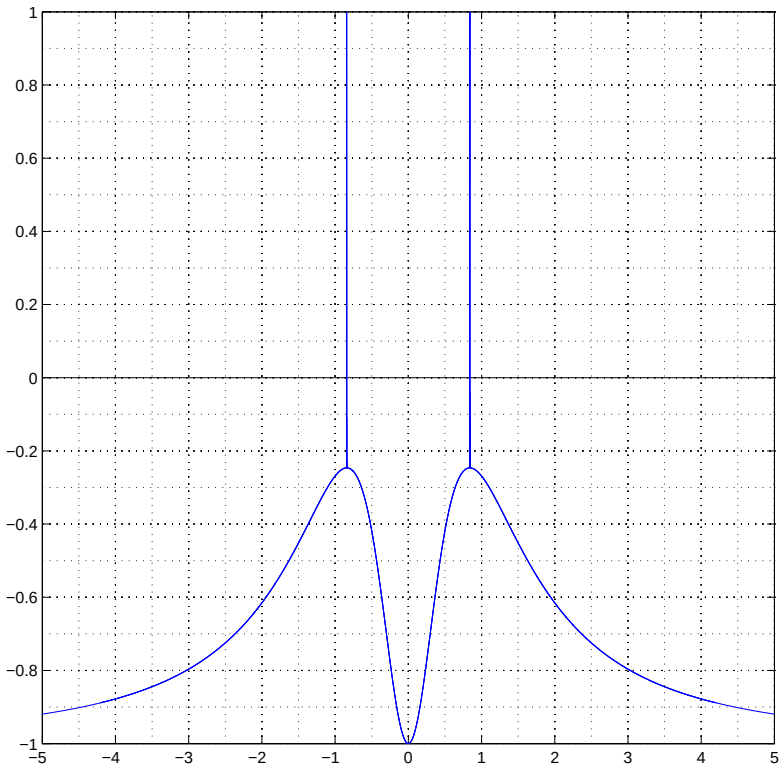


FIG. 8: Plot of psolqcap(r) , $f(\lambda) = p(\lambda)p(\lambda)^* - 1$; $a_3q^2 + b_3q + c_3 = 0$, hybrid#2: near-plane hill top with natural spike!

Now the reader might argue, that one could just look at the top of the hill and see if it is "breaking out" into a natural spike, or "smooth" indicative of an artifact there. So, let's consider another hybrid quadratic equation, `hybrid#2`, where we combine just 0.01% of the (a_1, b_1, c_1) deep space natural spike, with 99.99% of the (a_2, b_2, c_2) onplane hill top, to construct our hybrid (a_3, b_3, c_3), while at the same time we "thin out" the natural spike, reducing the setting of the parameter, $\beta_1 \cdot b_1$, from, $\beta_1 = 0.25$, to, $\beta_1 = 0.0001$.

```
  .
>> A3 = 0.0001*A1 + 0.9999*A2;
>> B3 = 0.0001*0.0001*B1 + 0.9999*1.0*B2;
>> C3 = 0.0001*C1 + 0.9999*C2;
>> psolqabc(A3,B3,C3)
>> r = sqrt(sqrt(trace(c*c')/trace(a*a')))
                                                    r = 0.8409
>> r1 = fzero(@psolq,[r-.1,r-10^-5])
                                                    r1 = 0.8407
>> r2 = fzero(@psolq,[r+10^-5,r+.1])
                                                    r2 = 0.8410
>> rr = -5*r:0.0001:5*r;
>> size(rr)

ans =

1 84086

>> for i = 1:84086
yy(i) = psolqcap(rr(i));
end
>> plot(rr,yy)
>> hold on
>> grid on
>> grid minor

>> q1 = r1*psolqv(r1)
                                                    q1 = 0.7759 0.1393 -0.0921 0.2773
>> q2 = r2*psolqv(r2)
                                                    q2 = 0.6727 -0.2164 0.1442 -0.4325
>> v1 = mulq(A3,mulq(q1,q1)) + mulq(B3,q1) + C3
                                                    v1 = 1.0e-12 *[ 0.3099 0.1634 -0.1081 0.3252 ]
>> v2 = mulq(A3,mulq(q2,q2)) + mulq(B3,q2) + C3
                                                    v2 = 1.0e-12 *[-0.2007 -0.0226 0.0152 -0.0454 ]
```

We still get a perfectly good solution with the natural spike hypothesis. The implicated solutions, q_1 and q_2, evaluate the quadratic expression, $aq^2 + bq + c$, to, 0, to the level of 10^{-13}. What happens if we remove the spike?

```
  Crush the spike...

>> p = psolqvh(r)
                                                    p = 0.8614 -0.0460 0.0311 -0.0926
>> pp = p*p'
                                                    pp = 0.7536
>> Vp = p
                                                    Vp = 0.8614 -0.0460 0.0311 -0.0926
>> Vp(1) = 0
                                                    Vp = 0 -0.0460 0.0311 -0.0926
>> nVp = Vp/sqrt(Vp*Vp')
                                                    nVp = 0 -0.4260 0.2879 -0.8577
>> m1 = [1,0,0,0] + nVp*sqrt(1/pp - 1)
                                                    m1 = 1.0000 -0.2436 0.1646 -0.4904
```

```
>> m2 = [1,0,0,0] - nVp*sqrt(1/pp - 1)
                                                    m2 = 1.0000 0.2436 -0.1646 0.4904
>> u1 = mulq(p,m1)
                                                    u1 = 0.7996 -0.2558 0.1729 -0.5150
>> u2 = mulq(p,m2)
                                                    u2 = 0.9231 0.1638 -0.1107 0.3298
>> q1 = r*u1
                                                    q1 = 0.6724 -0.2151 0.1454 -0.4331
>> q2 = r*u2
                                                    q2 = 0.7762 0.1377 -0.0931 0.2773
>> v1 = mulq(A3,mulq(q1,q1)) + mulq(B3,q1) + C3
                                                    v1 = 1.0e-03 *[ -0.1629 -0.1750 0.0466 0.4338 ]
>> v2 = mulq(A3,mulq(q2,q2)) + mulq(B3,q2) + C3
                                                    v2 = 1.0e-03 *[ 0.2787 0.0821 -0.4030 0.0246 ]
```

Hmmm...that hill top hypthesis is looking better indeed. We're now obtaining solutions, q_1 and q_2, that evaluate the $aq^2 + bq + c$ to, 0, to the level of 10^{-4}. The last time it was to level of 10^{-2}. While, we also notice, that while the natural spike solution does much better than the hill top here, it didn't do as well as last time, the precision slipped a little; last time we saw 10^{-14}, but now only, 10^{-13}. But, notice, the magnitudes are all moving in towards each other, $\lambda = r = 0.8409$, $\lambda_1 = \texttt{r1} = 0.8407$, $\lambda_2 = \texttt{r2} = 0.8410$, and the solutions, q_1 and q_2, from the two hypotheses, are converging towards each other. We can guess, that there's a point when it would be difficult to decide on which hypothesis is best. They would provide equally good competiting solutions. But, by then the question is moot, because the distinction between the different magnitudes would be at the level of precision where it didn't matter anymore. We mignt not know whether to call the solution "hill top natural spike" or "hill top artifical spike," but the solution would be the same, which ever way we decided to do the computation.

`Input Precision.` Of course, one might argue that this is an "*extreme*" example of a highly artificial problem deliberately constructed to test the limits of the calculation method. No one would ever encounter such a hybrid quaternion quadratic equation in practice. Who would be working on a problem that moved parameter points on a complex variable subplane, just off the plane, towards some random deep space point locations? It's inconceivable, isn't it? Many a software developer has thought this way, only to have some user come back later with a complaint that the code didn't work as expected, and a request for a bug fix. The user didn't do what the programmer thought the user should be doing.

In our onplane hill top example, we deliberately chose integer numbers to achieve the highest precision level we could for input parameters. We know the three (a, b, c)–parameters are exactly on the complex subplane. There is no error. Conceptually, the parameters fall on a complex plane. Actually, they fall on the complex plane. Things are as we intend. When we enter these whole number parameters into MATLAB, however, some calculation is done dividing, and there's a limit to the precision encoding numbers, so the perfect parameters "drift" a little from our ideal positions. Yet, the deviation is all the way down at the 15th decimal place. So, things still look good, and we are generally satisfied. If we ask MATLAB to show us what the numbers we entered in are, here is what we see by default;

```
>> A2
                                                    A2 = 1 0 0 0
>> B2
                                                    B2 = -1.4489 0.0776 -0.0517 0.1552
>> C2
                                                    C2 = 0.6853 -0.0746 0.0497 -0.1492
>>
```

MATLAB actually encodes the numbers to higher level of precision, so underneath there is more accuracy than shown, but it's often impractical to list all those decimal places all the time, so 4 decimals is usually convenient. Now suppose the user is satisfied with this precision level, and thinks to use 4 decimal places in his work, and so decides to "enter in" the parameters accurate to 4 decimals as shown, intending for these to be points on the same complex plane.

```
>> A4 = [1,0,0,0];
>> B4 = [-1.4489,0.0776,-0.0517,0.1552];
>> C4 = [0.6853,-0.0746,0.0497,-0.1492];
```

Having manually entered in the numbers as shown, the problem is then solved as before. First, under the natural spike hypothesis.

```
>> psolqabc(A4,B4,C4)
>> r = sqrt(sqrt(trace(c*c')/trace(a*a')))
                                                    r = 0.8409
>> r1 = fzero(@psolq,[r-.1,r-10^-5])
                                                    r1 = 0.8408
>> r2 = fzero(@psolq,[r+10^-5,r+.1])
                                                    r2 = 0.8409
>> q1 = r1*psolqv(r1)
                                                    q1 = 0.6726 -0.2162 0.1440 -0.4324
>> q2 = r2*psolqv(r2)
                                                    q2 = 0.7763 0.1386 -0.0923 0.2772
>> v1 = mulq(A4,mulq(q1,q1)) + mulq(B4,q1) + C4
                                                    v1 = 1.0e-12 *[ 0.6124 -0.0993 0.0660 -0.1985]
>> v2 = mulq(A4,mulq(q2,q2)) + mulq(B4,q2) + C4
                                                    v2 = 1.0e-12 *[-0.0981 -0.1708 0.1137 -0.3416]
```

Then under the artificial spike hypothesis...

```
Crush the spike..

>> p = psolqvh(r)
                                                    p = 0.8616 -0.0461 0.0307 -0.0921
>> Vp = p
                                                    Vp = 0.8616 -0.0461 0.0307 -0.0921
>> Vp(1) = 0
                                                    Vp = 0 -0.0461 0.0307 -0.0921
>> nVp = Vp/sqrt(Vp*Vp')
                                                    nVp = 0 -0.4285 0.2855 -0.8572
>> pp = p*p'
                                                    pp = 0.7539
>> m1 = [1,0,0,0] + nVp*sqrt(1/pp - 1)
                                                    m1 = 1.0000 -0.2449 0.1632 -0.4898
>> m2 = [1,0,0,0] - nVp*sqrt(1/pp - 1)
                                                    m2 = 1.0000 0.2449 -0.1632 0.4898
>> Q1 = r*mulq(p,m1)
                                                    Q1 = 0.6728 -0.2161 0.1440 -0.4323
>> Q2 = r*mulq(p,m2)
                                                    Q2 = 0.7761 0.1387 -0.0924 0.2774
>> V1 = mulq(A4,mulq(Q1,Q1)) + mulq(B4,Q1) + C4
                                                    V1 = 1.0e-03 *[ 0.1008 -0.0777 0.0541 -0.1488]
>> V2 = mulq(A4,mulq(Q2,Q2)) + mulq(B4,Q2) + C4
                                                    V2 = 1.0e-03 *[-0.1727 -0.0481 0.0262 -0.0920]
>>
```

Now we can understand the real issue. The user intended to put the three coefficients (a, b, c) on a complex subplane within the quaternion space. But, used numbers only accurate to 4 decimal places. So, they were'nt actually on the same complex plane. The numbers all have random drifts away from the "nearest plane's" ideal point locations, because of the rounding of the digits. So, believing that they are `onplane`, he crushes the spike, and finds solutions, `Q1` and, `Q2`, that evaluate the quadratic expression, $aq^2 + bq + c$, to, 0, only to the level of $1. \times 10^{-4}$, but that's exactly the precision he put in, when entering the numbers. Yet, when he checks the alternative hypothesis, he finds that the natural spike gives a superior solution, resolving the quadratic to level of $1. \times 10^{-13}$.

The user might still decide that the answer should be "*same magnitudes, different directions*," and therefore reject *the best numerical solution*, which tells us there are actually "*different magnitudes*," opting for the one he feels "conceptually" satisfied with, knowing that he entered in a limited precision, as it matches better with his "intent."

`Offplane Magic Factors`

Consider the case where the knowns, a, b, c, in the quadratic equation, $aq^2 + bq + c = 0$, are not all in the same complex subplane. This offplane situation will now be treated. We know, so far, that in the condition of degenerate magnitudes, the formulas for the two solutions, with same, λ, are;

$$q_1 = \lambda \cdot u_1 = \lambda \cdot p \cdot (1 + n\sqrt{|p|^{-2} - 1}) \tag{1.155}$$
$$q_2 = \lambda \cdot u_2 = \lambda \cdot p \cdot (1 - n\sqrt{|p|^{-2} - 1}) \tag{1.156}$$
$$\tag{1.157}$$

And, for the moment, the variable, n, is a pure imaginary quaternion, itself with unit magnitude, so, $|n| = 1$, $n^2 = -1$. But we don't know, yet, anything about the actual imaginary direction. It could, in principle, represent an infinity of possible solutions, $n = n_1 i + n_2 j + n_3 k, \;\; n_1, n_2, n_3 \in \mathbb{R}, \; n_1^2 + n_2^2 + n_3^2 = 1$. Or, it could be some finite number of possibilities. Writing the solutions this way, as a pair, with, $+n$ and $-n$, is a bit presumptuous, since in the case of an infinity of roots, we've obviously over-specified the solution. The q_2 are all contained in the q_1, in that case. But, it's still correct. And it's convenient, for the moment. We know the directions can't be the same, so, $u_1 \neq u_2$. Because when they are equal, we have, $p = (u_1 + u_2)/2 = (2u_1)/2 = u_1 = u_2$, which would mean that, $|p| = |u_1| = |u_2| = 1$, and we already know that, $|p| < 1$. Therefore, there must be *at least* two distinct roots, λu_1 and λu_2, when, $|p| < 1$, if any roots exist at all.

Note that the condition, "*Same magnitudes, same directions*," corresponds to that unique situation where the peak of the curve, $f(\lambda) = p(\lambda)p(\lambda)^* - 1$, just "touches" the $(f = 0)$–axis, at the critical point, $\lambda = \sqrt{|c|/|a|}$. When, $f(\lambda)$, moves above this axis, there are two distinct roots with different magnitudes, λ_1 and λ_2, and when, $f(\lambda)$, stays below this axis, there are again two distinct roots, but this time the magnitudes are the same, just the directions differ. The axis is the locus of the transition between these two states.

Now we need to figure out what the unit imaginary, n, is, to complete the solution. We know the result of the sum of the roots, $q_1 + q_2 = \lambda(u_1 + u_2) = 2\lambda p$, but we don't know, u_1 and u_2, individually. Nevertheless, we can evaluate the quadratic expression, using, $2\lambda p$, on the one hand, and using, $\lambda(u_1 + u_2)$, on the other, and just equate them.

$$a(\lambda(u_1 + u_2))^2 + b(\lambda(u_1 + u_2)) + c = a(2\lambda p)^2 + b(2\lambda p) + c \tag{1.158}$$
$$[a\lambda^2 u_1^2 + b\lambda u_1 + c] + [a\lambda^2 u_2^2 + b\lambda u_2 + c] - c + a\lambda^2(u_1 u_2 + u_2 u_1) = a(2\lambda p)^2 + b(2\lambda p) + c \tag{1.159}$$
$$0 + 0 - c + a\lambda^2(u_1 u_2 + u_2 u_1) = a(2\lambda p)^2 + b(2\lambda p) + c \tag{1.160}$$
$$a\lambda^2(u_1 u_2 + u_2 u_1) = a(2\lambda p)^2 + b(2\lambda p) + 2c \tag{1.161}$$
$$(u_1 u_2 + u_2 u_1)/2 = \lambda^{-2} a^{-1}(2a(\lambda p)^2 + b(\lambda p) + c) \tag{1.162}$$

We now have an expression for the *average of the products* of the unknown units, $(u_1 u_2 + u_2 u_1)/2$, on the l.h.s, in terms of a known quaternion, on the r.h.s. But, we also know, u_1 and u_2, in terms of a known quaternion, p, and some unknown imaginary unit, n. Plugging in these factors; Letting, $\gamma = \sqrt{|p|^{-2} - 1}$.

$$\begin{aligned}
(u_1 u_2 + u_2 u_1) &= p(1 + n\gamma)p(1 - n\gamma) + p(1 - n\gamma)p(1 + n\gamma) && (1.163)\\
&= (p + pn\gamma)(p - pn\gamma) + (p - pn\gamma)(p + pn\gamma) && (1.164)\\
&= (pp + pnp\gamma - ppn\gamma - pnpn\gamma^2) + (pp - pnp\gamma + ppn\gamma - pnpn\gamma^2) && (1.165)\\
&= (2pp - 2pnpn\gamma^2) && (1.166)
\end{aligned}$$

Therefore,

$$\begin{aligned}
(pn\gamma)^2 &= p^2 - (u_1 u_2 + u_2 u_1)/2 && (1.167)\\
&= p^2 - \lambda^{-2} a^{-1}(2a(\lambda p)^2 + b(\lambda p) + c) && (1.168)\\
&= -\lambda^{-2} a^{-1}(a(\lambda p)^2 + (b(\lambda p) + c) && (1.169)
\end{aligned}$$

Rearranging this equation we have,

$$\lambda^2 a(pn\gamma)^2 + (a(\lambda p)^2 + (b(\lambda p) + c) = 0 \tag{1.170}$$
$$AQ^2 + C = 0 \;\; , \qquad Q = pn\sqrt{|p|^{-2} - 1}, \;\; A = \lambda^2 a, \;\; C = (a(\lambda p)^2 + (b(\lambda p) + c) \tag{1.171}$$

This has the form of the equation, $aq^2 + bq + c = 0$, with, $b = 0$, whose solutions are derived below in APPENDIX C. We put the derivations there, because *one-step* methods solve these types of equations. But, we still elect to solve in

the style of this paper, using our separation, $q = \lambda \cdot p$, to see the correspondences. We repeat the final results of those solutions here;

$$AQ^2 + C = 0 \tag{1.172}$$

$$\texttt{Case 1: } A^*C \in \mathbb{R},\ A^*C > 0. : \quad Q = \sqrt{|C|/|A|} \cdot (m_1 i + m_2 j + m_3 k) \tag{1.173}$$

$$\texttt{Case 2: } A^*C \in \mathbb{R},\ A^*C < 0. : \quad Q = \pm\sqrt{|C|/|A|} \cdot 1 \tag{1.174}$$

$$\texttt{Case 3: } A^*C \notin \mathbb{R},\ S(A^*C) = 0. : \quad Q = \pm\sqrt{|C|/|A|} \cdot (1 - A^*C/|A^*C|)/\sqrt{2} \tag{1.175}$$

$$\texttt{Case 4: } A^*C \notin \mathbb{R},\ S(A^*C) \neq 0. : \quad Q = \pm\sqrt{|C|/|A|} \cdot (1 - A^*C/|A^*C|/)/\sqrt{2 - (A^*C + (A^*C)^*)/|A^*C|} \tag{1.176}$$

$$A^*C = \lambda^2 a^* (a(\lambda p)^2 + b(\lambda p) + c) \tag{1.177}$$

$$\frac{|C|}{|A|} = \frac{|(a(\lambda p)^2 + b(\lambda p) + c)|}{\lambda^2 |a|} = \frac{|(a(\lambda p)^2 + b(\lambda p) + c)|}{|c|} \ , \qquad \lambda^2 = |c|/|a| \tag{1.178}$$

Putting this all together, we can finally write down the solution to the quadratic equation, $aq^2 + bq + c = 0$, given, $b \neq 0$, for the case where we have *Same magnitudes, different directions*. There are 4 sub-cases to consider. In sub-case 1, we have an infinite number of roots. The magnitude is degenerate, but to an infinity of general quaternion directions. All other sub-cases have just two distinct roots. The sub-case 3 and sub-case 4 could be combined together into a single sub-case, using only 4. However, it helps to see the distinctions this way.

$$q_1 = \lambda \cdot u_1 = \lambda \cdot \left(p - u \cdot \sqrt{\frac{|a(\lambda p)^2 + b(\lambda p) + c|}{|c|}} \right) \tag{1.179}$$

$$q_2 = \lambda \cdot u_2 = \lambda \cdot \left(p + u \cdot \sqrt{\frac{|a(\lambda p)^2 + b(\lambda p) + c|}{|c|}} \right) \tag{1.180}$$

$$w = \lambda^2 a^* (a(\lambda p)^2 + b(\lambda p) + c) \ , \qquad \lambda = \sqrt{\frac{|c|}{|a|}} \tag{1.181}$$

$$\text{where, } u = u(w) : \tag{1.182}$$

$$\texttt{Case 1: } w \in \mathbb{R},\ w > 0. : \quad u = (m_1 i + m_2 j + m_3 k) \tag{1.183}$$

$$\texttt{Case 2: } w \in \mathbb{R},\ w < 0. : \quad u = 1 \tag{1.184}$$

$$\texttt{Case 3: } w \notin \mathbb{R},\ S(w) = 0. : \quad u = (1 - w/|w|)/\sqrt{2} \tag{1.185}$$

$$\texttt{Case 4: } w \notin \mathbb{R},\ S(w) \neq 0. : \quad u = (1 - w/|w|/)/\sqrt{2 - (w + w^*)/|w|} \tag{1.186}$$

So, what we learn from all this, is that the values of *two averages* are required to establish the complete solution, to the degenerate case. We need the average of the sum of the solutions, $(q_1 + q_2)/2$, and the average of the products of the solutions, $(q_1 q_2 + q_2 q_1)/2$. Since the magnitudes are the same, we extract the magnitude, λ, and then we just need to find the average of the direction units, $(u_1 + u_2)/2$, and the average of their products, $(u_1 u_2 + u_2 u_1)/2$. With this information in hand, we can calculate both roots, q_1 and q_2, separating them out from their average measure.

$$q = \lambda \cdot p \cdot \left(1 \pm \frac{p^* u}{|p|^2} \cdot \sqrt{\frac{|a(\lambda p)^2 + b(\lambda p) + c|}{|c|}} \right) \tag{1.187}$$

$$q = \lambda \cdot p \cdot \left(1 \pm n \cdot \sqrt{|p|^{-2} - 1} \right) \tag{1.188}$$

$$n \cdot \sqrt{|p|^{-2} - 1} = \frac{p^* u}{|p|^2} \cdot \sqrt{\frac{|a(\lambda p)^2 + b(\lambda p) + c|}{|c|}} \tag{1.189}$$

Of course, it is well known from abelian algebra, that the quadratic equation is determined by the sum, $z_1 + z_2$, and the product, $z_1 z_2$, of its roots. But, in non-abelian algebra, what is the product of the roots? There are now two products, $q_1 q_2$ and $q_2 q_1$. It is logical, then, that the method requires an average of the two possibilities, to find the solution here. There would be no reason to pick one over the other, nor to weight one more heavily than the other,

so in the case of degeneracy, we expect equal weighting of the parts of solution. This is what we find, algebraically. Intuition and algebra are in agreement.

We've written our final solutions in terms of a new unit quaternion, u. This is a regular quaternion, with unit magnitude, $|u| = 1$, not generally a pure imaginary unit, i.e., $u^2 \neq -1$. But, our orignal magic factor required a pure imaginary unit, n, with, $n^2 = -1$, so, we'd like to relate the two. Writing formulas with, u, helps to save some space and clutter. But, our original thinking was in terms of, n. So, to complete this bit of analysis, we extract the provisional direction unit, p, which represents the average of the true direction units, from the final solution factors, and relate the n and u unit quaternions to see the connexion. The magic is then seen to be that, n, is proportional to the product of the conjugate of the average unit, p^*, and the unit, u, with some constant factors scaling the result back to unit magnitude, so that, $|n| = 1$.

`Paradox at Infinity.` Now we have to deal with a very subtle matter. It concerns the infinity roots case. Perhaps, the reader noticed already, that there is something strange about all this, when it comes to counting the roots. We *assumed*, that when, $|p| < 1$, that means that there's a degeneracy in the magnitude, and the p is then *the average* of the unit directions. Our justification, for this assumption, comes from the similar situation in the complex variable example, where we have actually proved this algebraically. Then, we wrote the average, $p = (u_1 + u_2)/2$, as if there could only be just two direction units, as in the complex variable case, but then after all the calculations, found that in one case there's actually an infinite number of directions involved. So the paradox is that our initial hypothesis that two things make up an average, used to derive a result, finds there are infinitely many things after all, seeming to contradict the initial hypothesis. So, how do we find the average of an infinite number of unit directions?

I will give the argument that a Physicist would when doing mathematics, finding infinities in the results, renormalizing by changing the order of computations, thus resolving the apparent paradox, and leave it up to the true mathematicians to improve on the answer.

In the infinity case, $u = (m_1 i + m_2 j + m_3 k),\ m_1, m_2, m_3 \in \mathbb{R},\ m_1^2 + m_2^2 + m_3^2 = 1$, we can divide the infinite cleanly into two sets, $u_1 = p + (m_1 i + m_2 j + m_3 k)\beta$, and, $u_2 = p - (m_1 i + m_2 j + m_3 k)\beta$, making sure never to count the same unit twice. Every unit has a related unit, exactly opposite it on the unit sphere through the origin. So, just pair up the units. Then, we take the average of opposite pairs, we get, $(u_1 + u_2)/2 = (p + u\beta + p - u\beta)/2 = p$; the average of N such unit pairs, still gives, p, so let, $N \to \infty$, and this still calculates to, p, however large we make, N. Hence, the average of the infinity of roots is still just the average of any two opposite units, which is, p. Q.E.D. Physics Style.

`Closed form for direction AND magnitude!` Note that in this degenerate case, when, $|p(\lambda)| < 1,\ \forall\lambda$, we have found a completely closed solution. Both magnitude and direction are fully determined without any searching or numerical solution methods required. We know the peak of the function is at the critical point. So, we plug in that λ to see if the peak is above or below, $|p(\lambda = \sqrt{|c|/|a|})| < 1$? If, $|p| < 1$, the magnitude is degenerate and given by, $\lambda = \sqrt{|c|/|a|}$. We plug that magnitude into the formula for provisional direction unit, $p(\lambda)$, to obtain the average of the true unit directions. Once we've calculated this, p, we plug in the values of all the parameters, λ, p, a, b, c, using the magic factor formulas to determine the two solutions, q_1 and q_2. The only thing to be aware of is the computer or calculator finite precision limits, in calculating with the formulas near to the critical point $\lambda = \sqrt{|c|/|a|}$. Special care is required to avoid the NaN, making use of *L'Hôspital's rule* as appropriate.

`A Simple Check.` We just saw an `onplane` situation, which only required a simple "fix up" factor, $\sqrt{|p|^{-2} - 1}$, and the vector part of the provisional unit direction, p, to determine the magic factors required to obtain the solutions in the previous example. Here, in the `offplane` case, we have completely different looking magic than before. An obvious question is, can this possibly give us the same answer we got previously? The expressions look so unrelated. We simply have to check. Just out of curiousity. Even though the new magic is for offplane, we'll first confirm that it indeed gives the same result as the onplane magic, when applied to a complex subplane example.

```
>> global a b c
>> A = [1,0,0,0];
>> B = -( 5*([4,0,0,0] - 3*[0,3,-2,6]/7)/5 + 5*([12,0,0,0] + 5*[0,3,-2,6]/7)/13 );
>> C = mulq( 5*([4,0,0,0] - 3*[0,3,-2,6]/7)/5, 5*([12,0,0,0] + 5*[0,3,-2,6]/7)/13 );
>> psolqabc(A,B,C)
>> r = sqrt(sqrt(trace(c*c')/trace(a*a')))
                                                r = 5
```

```
  Crush the spike....

>> p = psolqvh(r)
                                                    p = 0.8615 -0.0462 0.0308 -0.0923
>> q = r*p
                                                    q = 4.3077 -0.2308 0.1538 -0.4615

>> E = mulq(A,mulq(q,q)) + mulq(B,q) + C % compute:  E = a(rp)^2 + b(rp) + c

E = 5.9645 -0.6492 0.4328 -1.2984

>> cA = [A(1),-A(2),-A(3),-A(4)]
                                                    cA = 1 0 0 0
>> w = r^2*mulq(cA,E)
                                                    w = 149.1124 -16.2299 10.8199 -32.4598
>> nw = w/sqrt(w*w')
                                                    nw = 0.9692 -0.1055 0.0703 -0.2110
>> u = ([1,0,0,0] - nw)/sqrt(2 - 2*nw(1))
                                                    u = 0.1240 0.4253 -0.2835 0.8505
>> q1 = r*(p - u*sqrt(sqrt((E*E')/(C*C'))))
                                                    q1 = 4.0000 -1.2857 0.8571 -2.5714
>> q2 = r*(p + u*sqrt(sqrt((E*E')/(C*C'))))
                                                    q2 = 4.6154 0.8242 -0.5495 1.6484
>> v1 = mulq(A,mulq(q1,q1)) + mulq(B,q1) + C
                                                    v1 = 1.0e-14 *[0.3553 0.4441 -0.1776 0.8882]
>> v2 = mulq(A,mulq(q2,q2)) + mulq(B,q2) + C
                                                    v2 = 1.0e-13 *[-0.1776 -0.0444 0.0222 -0.0711]
>>
```

Starting out as before, we enter the input parameters, `A,B,C`, then initialize the `global a b c`, with a call to `psolqabc(A,B,C)`, and find the degenerate magnitude, $\lambda = r = 5$. Using *L'Hôspital's rule* to get `p(r=5)`, we then proceed to compute along the new path. Calc the average root, `q = rp`, and use this in the quadratic expression to find the "residual" quaternion, `E = a(rp)^2 + b(rp) + c`. Construct the conjugate, `cA = A`*, and make the variable, `w` $= \lambda^2 a^*(a(\lambda p)^2 + b(\lambda p) + c)$. Looking at the components of `w`, we see that, `w` $\notin \mathbb{R}$, and that, `S(w)` $\neq 0$, so the relevant solution is "Case 4." This requires us to compute the unit, `nw = w/|w|`, and read out the scalar part, `nw(1)`, to compute, `u`. Finally, we can build the solutions, q_1 and q_2, selecting the appropriate sign, $\pm u$, to match our previous labeling, then, we can `Verify results...` we find our new solutions evaluate the quadratic expression, aq^2+bq+c, to, 0, to the level of 10^{-14}, just a hair better than the previous `onplane magic factor` results!

The `offplane magic factor` solution method works, just as well as the `onplane magic` did before. Perhaps, even a little better. But, this is just one test example. What we really need, is a test case for a true offplane example. So, now we're going to deliberately construct a more relevant offplane quadratic, to test the new magic factors. To do this, we make use of the same pythagorean triples, $3^2 + 4^2 = 5^2$, $5^2 + 12^2 = 13^2$, and quadruple square, $2^2 + 3^2 + 6^2 = 7^2$, which we used before. Then, we'll use the *direct equations* implied by the quadratic equation's two roots, to construct the coefficients, (a, b, c).

$$a = a \tag{1.190}$$
$$b = -a(q_1^2 - q_2^2)(q_1^* - q_2^*)/|q_1 - q_2|^2 \tag{1.191}$$
$$c = +a(q_1 - q_2)q_2(q_1^* - q_2^*)q_1/|q_1 - q_2|^2 \tag{1.192}$$

This time, however, we set up two different directions, using the same squares, by just rearranging the numbers. So, we set, $n_1 = (6i - 2j + 3k)/7$, and, $n_2 = (2i + 3j - 6k)/7$, (thinking of these like Heaviside-Gibbs vectors, we can see the "dot product," $n_1 \cdot n_2 = (12 - 6 - 18)/49 = -12/49 \neq \pm 1$, so the vector directions are significantly non-aligned), then we can construct two candidate roots, $Q_1 = 5 \cdot (4 - 3n_1)/5$, and, $Q_2 = 5 \cdot (5 + 12n_2)/13$.

$$33293.q^2 - (197197 + 46566.i - 27846.j - 97227.k).q + (574340 + 483210.i + 139230.j - 331695.k) \tag{1.193}$$

So, we then get the equation shown above, with perfect integer quaternion coefficients, for the three, (a, b, c), although the numbers that show up are a bit large, on the order of, 10^5. Since floating point calculations are limited by number of significant figures, we expect to see fewer decimal places accuracy to the right of the period than before

in the working out and verification of solutions. We could re-scale this equation by dividing by the lead parameter, a, but we'll leave that as an exercise for the reader. We keep the somewhat unwieldly large numbers, since the solution method really shouldn't care about the size of the numbers we use, anyway. Let's set up, and solve this equation.

The True Offplane:

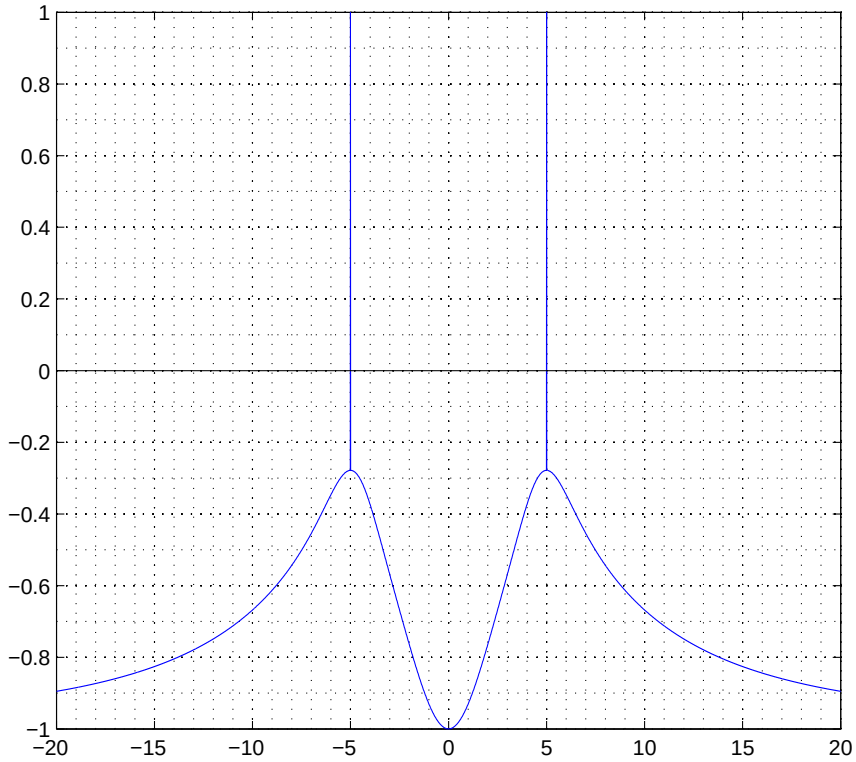


FIG. 9: Plot of psolqcap(r) , $f(\lambda) = p(\lambda)p(\lambda)^* - 1$; $aq^2 + bq + c = 0$, offplane same magnitudes, different directions
$33293.q^2 - (197197 + 46566.i - 27846.j - 97227.k).q + (574340 + 483210.i + 139230.j - 331695.k)$

```
>> clear all
>> hold off
>> global a b c
>> A = [33293,0,0,0]                                    A = 33293 0 0 0
>> B = -[197197,46566,-27846,-97227]                    B = -197197 -46566 27846 97227
>> C = [574340,483210,139230,-331695]                   C = 574340 483210 139230 -331695
>> psolqabc(A,B,C)
>> r = sqrt(sqrt(trace(c*c')/trace(a*a')))              r = 5
>> rr = -20:.01:20;
>> size(rr)                                             ans = 1 4001
>> for i = 1:4001
yy(i)=psolqcap(rr(i));
end
>> plot(rr,yy)
>> hold on
>> plot([-20 20],[0 0],'k-')
>> grid on
>> grid minor
>>
```

```
>> psolqv(r)                                        ans = Inf 0 -Inf Inf
>> psolqv(r+10^-5)                                  ans = 0.5923 -0.1253 0.2835 -0.5242
>> psolqv(r-10^-5)                                  ans = 0.5923 -0.1253 0.2835 -0.5242
>> p = psolqvh(r)                                     p = 0.5923 -0.1253 0.2835 -0.5242
>> pp = p*p'                                         pp = 0.7217
>> q = r*p                                            q = 2.9615 -0.6264 1.4176 -2.6209
>> E = mulq(A,mulq(q,q)) + mulq(B,q) + C          E = 1.0e+05 * 1.5986 1.3449 0.3875 -0.9232
>> cA = [A(1),-A(2),-A(3),-A(4)]                     cA = 33293 0 0 0
>> w = r^2*mulq(cA,E)                             w = 1.0e+11 * 1.3306 1.1194 0.3225 -0.7684
>> nw = w/sqrt(w*w')                                 nw = 0.6900 0.5806 0.1673 -0.3985
>> u = ([1,0,0,0] - nw)/sqrt(2 - 2*nw(1))             u = 0.3937 -0.7374 -0.2125 0.5062
>> u1 = p + u*sqrt(sqrt((E*E')/(C*C')))              u1 = 0.8000 -0.5143 0.1714 -0.2571
>> u2 = p - u*sqrt(sqrt((E*E')/(C*C')))              u2 = 0.3846 0.2637 0.3956 -0.7912
>> q1 = r*u1                                         q1 = 4.0000 -2.5714 0.8571 -1.2857
>> q2 = r*u2                                         q2 = 1.9231 1.3187 1.9780 -3.9560
>> Q1 = 5*([4,0,0,0] - 3*[0,6,-2,3]/7)/5             Q1 = 4.0000 -2.5714 0.8571 -1.2857
>> Q2 = 5*([5,0,0,0] + 12*[0,2,3,-6]/7)/13           Q2 = 1.9231 1.3187 1.9780 -3.9560
>> v1 = mulq(A,mulq(q1,q1)) + mulq(B,q1) + C          v1 = 1.0e-09 * 0 0 -0.1164 -0.1164
>> v2 = mulq(A,mulq(q2,q2)) + mulq(B,q2) + C          v2 = 1.0e-09 * 0.3492 0.1164 0.0873 -0.0582
>> V1 = mulq(A,mulq(Q1,Q1)) + mulq(B,Q1) + C          V1 = 1.0e-10 * 0 0 -0.2910 -0.5821
>> V2 = mulq(A,mulq(Q2,Q2)) + mulq(B,Q2) + C          V2 = 1.0e-09 * 0 0 0 -0.1164
>>
```

So here we see that the solutions we "found," q_1 and q_2, evaluate the quadratic expression, $aq^2 + bq + c$, to, 0, at the level of, 10^{-10}, while the solutions we "put in," Q_1 and Q_2, evaluate this expression to, 10^{-11}. Although seemingly less precise than our previous results where we got down to level, 10^{-15}, remember we put in large numbers on the order of, 10^5, which consumes some of our significant digits. So, we're really at the same level of precision in terms of significant digits as before. We do get lucky here, though, and find a few actual 0's in the verification results. If we re-scale the quadratic equation, dividing throughout by the lead, $a = 33293$, to make the initial input parameters closer to unity, then the verification results generally jump down to 10^{-15}, but the graph changes, looses it's spikes, and we miss out on a few lucky 0s. The ability to see some of these finer details of the quaternion quadratic equation depends, to a certain extent, on the mathematical software used to explore these solutions. When we switch from MATLAB to OCTAVE, we find slightly different numbers in the less significant digits, and some graphs miss plotting out the spikes, and so on. So, these quaternion quadratic problems are a good test of the numerical software's ability to represent the state of things, and reveal detail that is there to see.

`A final offplane.`

One last example, constructed from a modified version of a quadratic equation given by Hamilton, with smaller input numbers, for this *same magnitudes, different directions* case, will be considered here.

$$(7 + 5i - 3j - k)q^2 + (-2 + 6i + 10j - 14k)q + (125 + 75i + 50j + 50k) = 0 \tag{1.194}$$

A glance at the parameters, (a, b, c), tells us immediately, that the vector parts of these quaternions are linearly independent of each other. Let's run this example to see what we get.

```
>> clear all
>> global a b c
>> A = [7,5,-3,-1];
>> B = [-2,6,10,-14];
>> C = [125,75,50,50];
>> psolqabc(A,B,C)
>> r = sqrt(sqrt(trace(c*c')/trace(a*a')))

r =

4.2045

>> rr = -10:0.001:10;
```

```
>> size(rr)

ans =

1 20001

>> for i = 1:20001
yy(i)=psolqcap(rr(i));
end
>> plot(rr,yy)
>> hold on
>> plot([-10 10],[0 0],'k-')
>> grid on
>> grid minor
```

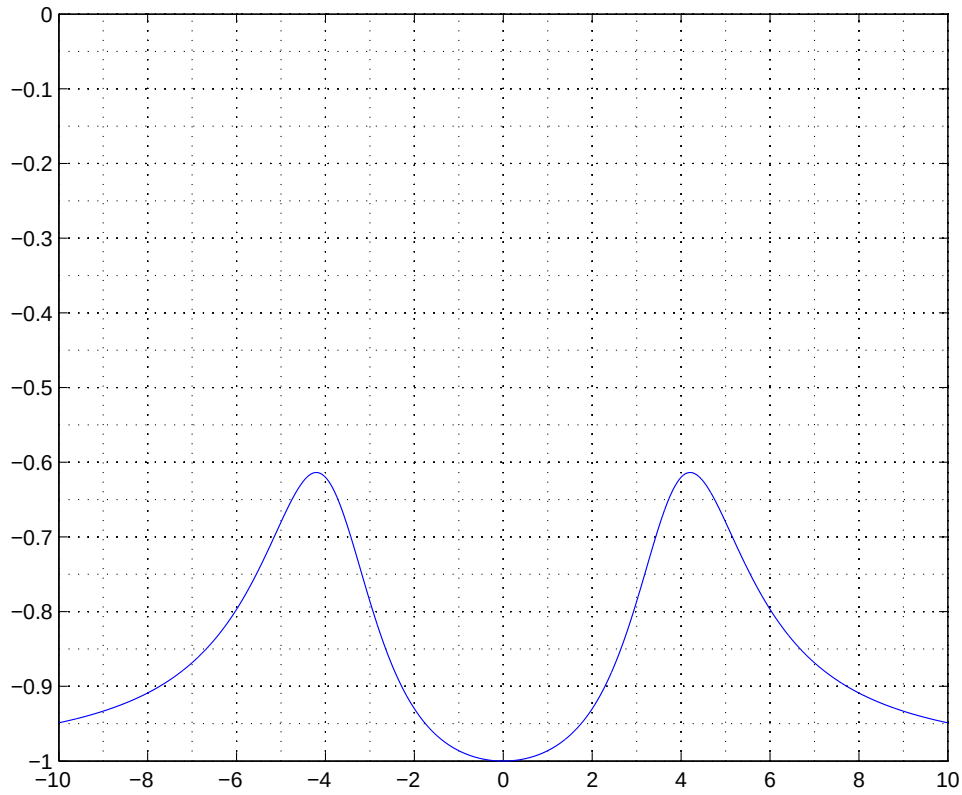


FIG. 10: Plot of psolqcap(r) , $f(\lambda) = p(\lambda)p(\lambda)^* - 1$; $aq^2 + bq + c = 0$, An offplane quadratic hill top with rough peak

$$(7 + 5i - 3j - k)q^2 + (-2 + 6i + 10j - 14k)q + (125 + 75i + 50j + 50k) = 0 \tag{1.195}$$

There are no visible spikes here, neither natural spike, nor atrifact. However, the top of the hill is still not smooth enough for reading data directly. If we test the peak, and nearby points on either side, we can verify the blip of artifact is still present, even though not visible to the eye.

```
Test the peak...

>> p = psolqv(r)
```

```
                                                    p = -0.0882 0.3916 0.0973 -0.0023
>> p = psolqv(r-10^-5)
                                                    p = 0.0000 0.5742 -0.0000 0.2378
>> p = psolqv(r+10^-5)
                                                    p = -0.0000 0.5742 0.0000 0.2378
>>
```

A bit of a jump there, where we know things should be smooth. So, we again ask our good friend *L'Hôspital*, to read under the bump. Then, we calculate all the details, find, q_1, and, q_2, and..`Verify results...`

```
>> p = psolqvh(r)
                                                    p = 0.0000 0.5742 -0.0000 0.2378
>> pp = p*p'
                                                    pp = 0.3863
>> q = r*p
                                                    q = 0.0000 2.4142 -0.0000 1.0000
>> E = mulq(A,mulq(q,q)) + mulq(B,q) + C
                                                    E = 76.7157 46.0294 30.6863 30.6863
>> cA = [A(1),-A(2),-A(3),-A(4)]
                                                    cA = 7 -5 3 1
>> w = r^2*mulq(cA,E)
                                                    w = 1.0e+04 *[ 1.1392 0.0000 1.1392 0]
>> nw = w/sqrt(w*w')
                                                    nw = 0.7071 0.0000 0.7071 0
>> u = ([1,0,0,0] - nw)/sqrt(2 - 2*nw(1))
                                                    u = 0.3827 -0.0000 -0.9239 0
>> u1 = p + u*sqrt(sqrt((E*E')/(C*C')))
                                                    u1 = 0.2998 0.5742 -0.7238 0.2378
>> u2 = p - u*sqrt(sqrt((E*E')/(C*C')))
                                                    u2 = -0.2998 0.5742 0.7238 0.2378
>> q1 = r*u1
                                                    q1 = 1.2605 2.4142 -3.0431 1.0000
>> q2 = r*u2
                                                    q2 = -1.2605 2.4142 3.0431 1.0000
>> v1 = mulq(A,mulq(q1,q1)) + mulq(B,q1) + C
                                                    v1 = 1.0e-13 *[ 0.5684 0.7105 0.0711 0.0711]
>> v2 = mulq(A,mulq(q2,q2)) + mulq(B,q2) + C
                                                    v2 = 1.0e-13 *[ 0.2842 -0.2842 0.0711 0.4263]
>>
```

The two solutions, `q1` and `q2`, evaluate the quadratic, $aq^2 + bq + c$, to, 0, to the level of 10^{-14}. The equation, plus solutions, are as follows;

$$(7 + 5i - 3j - k)q^2 + (-2 + 6i + 10j - 14k)q + (125 + 75i + 50j + 50k) = 0 \tag{1.196}$$

$$q_1 = +1.2605 + 2.4142i - 3.0431j + 1.0000k\,, \qquad \lambda = 4.2045 \tag{1.197}$$
$$q_2 = -1.2605 + 2.4142i + 3.0431j + 1.0000k\,, \qquad \lambda = 4.2045 \tag{1.198}$$

Well, having seen a few examples, and studied the plots, convincing ourself that the method works, it's time to prove that it actually always does work. We can't prove anything with numerical software programs, because there's always doubt, that the small numbers appearing at verification might actually not be precision errors, but represent a true deviation from absolute 0, in some way, by the algebra itself. So we get 1.0×10^{-15}, but is that really $= 0$?

It't time to take a trek through the non-abelian wonderland!

`Proving:`

Consider the quadratic equation, $aq^2 + bq + c = 0$, with two roots, $q_1 = m \cdot u_1$ and $q_2 = m \cdot u_2$, and the two direct equations constructed from each root. Here, $|u_1| = 1,\ |u_2| = 1;\ \therefore\ 1/u_1 = u_1^*,\ 1/u_2 = u_2^*$; and, $u_1 \neq u_2$.

$$aq_1^2 + bq_1 + c = 0 \tag{1.199}$$
$$aq_2^2 + bq_2 + c = 0 \tag{1.200}$$

$$am^2u_1^2 + bmu_1 + c = 0 \tag{1.201}$$
$$am^2u_2^2 + bmu_2 + c = 0 \tag{1.202}$$
$$am^2(u_1^2 - u_2^2) + bm(u_1 - u_2) = 0 \tag{1.203}$$
$$b = -am(u_1^2 - u_2^2)(u_1^* - u_2^*)/|u_1 - u_2|^2 \tag{1.204}$$

$$am^2u_1 + bm + cu_1^* = 0 \tag{1.205}$$
$$am^2u_2 + bm + cu_2^* = 0 \tag{1.206}$$
$$am^2(u_1 - u_2) + c(u_1^* - u_2^*) = 0 \tag{1.207}$$
$$c = -am^2(u_1 - u_2)^2/|u_1 - u_2|^2 \tag{1.208}$$

$$\text{Let,}\quad s = u_1 + u_2,\ \ d = u_1 - u_2 \tag{1.209}$$
$$sd = (u_1 + u_2)(u_1 - u_2) = u_1^2 - u_2^2 + u_2u_1 - u_1u_2 \tag{1.210}$$
$$ds = (u_1 - u_2)(u_1 + u_2) = u_1^2 - u_2^2 - u_2u_1 + u_1u_2 \tag{1.211}$$
$$sd + ds = 2(u_1^2 - u_2^2) \tag{1.212}$$
$$b = -(1/2)am(sd + ds)d^*/|d|^2 \tag{1.213}$$
$$c = -am^2d^2/|d|^2 \tag{1.214}$$

By this procedure, we establish the relationship between the three parameters, (a, b, c), of the quadratic equation, and it's two roots, $q_1 = m \cdot u_1$ and $q_2 = m \cdot u_2$, where we've represented the sum of the units by, $s = u_1 + u_2$, and the difference of the units by, $d = u_1 - u_2$, to abbreviate expression manipulations. Now consider the formula under test, our usual *provisional* unit direction function.

$$p(\lambda) = \frac{\lambda^5(\lambda^4a^*aa^* - c^*ac^*)(cb^* - \lambda^2ba^*)a + \lambda(c^*cc^* - \lambda^4a^*ca^*)(cb^* - \lambda^2ba^*)c}{(\lambda^4|a|^2 - |c|^2) \cdot |\lambda^2a - c|^2 \cdot |\lambda^2a + c|^2} \tag{1.215}$$

We now replace the parameters, (a, b, c), by their root equivalents,

$$(\lambda^4a^*aa^* - c^*ac^*) = (\lambda^4a^*aa^* - (-am^2d^2/|d|^2)^* \cdot a \cdot (-am^2d^2/|d|^2)^*) \tag{1.216}$$
$$= (\lambda^4a^*aa^* - (m^2(d^*)^2a^*/|d|^2) \cdot a \cdot (m^2(d^*)^2a^*/|d|^2)) \tag{1.217}$$
$$= (\lambda^4a^*aa^* - m^4/|d|^4((d^*)^2 \cdot a^*a \cdot (d^*)^2a^*)) \tag{1.218}$$
$$= (\lambda^4a^*aa^* - m^4/|d|^4((d^*)^4 \cdot a^*aa^*)) \tag{1.219}$$
$$= (\lambda^4 - m^4(d^*)^4/|d|^4) \cdot a^*aa^* \tag{1.220}$$
$$= 1/|d|^4 \cdot (\lambda^4|d|^4 - m^4(d^*)^4) \cdot a^*aa^* \tag{1.221}$$
$$= 1/|d|^4 \cdot (\lambda^4(d^2)(d^*)^2 - m^4(d^*)^4) \cdot a^*aa^* \tag{1.222}$$
$$= (d^*)^2/|d|^4 \cdot (\lambda^4d^2 - m^4(d^*)^2) \cdot a^*aa^* \tag{1.223}$$
$$= 1/|d|^4 \cdot (\lambda^4d^2 - m^4(d^*)^2) \cdot (d^*)^2 \cdot a^*aa^* \tag{1.224}$$

$$(c^*cc^* - \lambda^4a^*ca^*) = ((-am^2d^2/|d|^2)^* \cdot (-am^2d^2/|d|^2) \cdot (-am^2d^2/|d|^2)^* - \lambda^4a^* \cdot (-am^2d^2/|d|^2) \cdot a^*) \tag{1.225}$$
$$= (-m^6/|d|^6 \cdot (d^*)^2a^* \cdot ad^2 \cdot (d^*)^2a^* + \lambda^4m^2/|d|^2 \cdot a^* \cdot ad^2 \cdot a^*) \tag{1.226}$$
$$= (-m^6/|d|^6 \cdot (d^*)^2 \cdot d^2 \cdot (d^*)^2 \cdot a^*aa^* + \lambda^4m^2/|d|^2 \cdot d^2 \cdot a^*aa^*) \tag{1.227}$$
$$= (-m^6/|d|^6 \cdot (d^*)^2 \cdot |d|^4 + \lambda^4m^2/|d|^2 \cdot d^2) \cdot a^*aa^* \tag{1.228}$$
$$= m^2/|d|^2 \cdot (\lambda^4d^2 - m^4 \cdot (d^*)^2) \cdot a^*aa^* \tag{1.229}$$
$$\tag{1.230}$$

$$p(\lambda) \quad = \begin{pmatrix} \lambda^5 \cdot [\ 1/|d|^4 \cdot (\lambda^4 d^2 - m^4 (d^*)^2) \cdot (d^*)^2 \cdot a^* a a^*\ ] \cdot (cb^* - \lambda^2 b a^*) a \\ +\lambda \cdot [\ m^2/|d|^2 \cdot (\lambda^4 d^2 - m^4 (d^*)^2) \cdot a^* a a^*\ ] \cdot (cb^* - \lambda^2 b a^*) c \\ \hline (\lambda^4 |a|^2 - |c|^2) \cdot |\lambda^2 a - c|^2 \cdot |\lambda^2 a + c|^2 \\ \end{pmatrix} \tag{1.231}$$

$$= (\lambda^4 d^2 - m^4 (d^*)^2) \cdot \begin{pmatrix} \lambda^5 \cdot [\ 1/|d|^4 \cdot (d^*)^2 \cdot a^* a a^*\ ] \cdot (cb^* - \lambda^2 b a^*) a \\ +\lambda \cdot [\ m^2/|d|^2 \cdot a^* a a^*\ ] \cdot (cb^* - \lambda^2 b a^*) c \\ \hline (\lambda^4 |a|^2 - |c|^2) \cdot |\lambda^2 a - c|^2 \cdot |\lambda^2 a + c|^2 \\ \end{pmatrix} \tag{1.232}$$

Working on the denominator factors,

$$|\lambda^2 a - c|^2 \quad = |\lambda^2 a + a m^2 d^2/|d|^2|^2 = |a|^2 \cdot |\lambda^2 + m^2 d^2/|d|^2|^2 = |a|^2 \cdot |\lambda^2 + m^2 (d^*)^2/|d|^2|^2 \tag{1.233}$$
$$= |a|^2 |d|^{-4} \cdot |\lambda^2 |d|^2 + m^2 (d^*)^2|^2 \tag{1.234}$$
$$= |a|^2 |d|^{-4} \cdot |\lambda^2 d d^* + m^2 (d^*)^2|^2 \tag{1.235}$$
$$= |a|^2 |d|^{-4} |d^*|^2 \cdot |\lambda^2 d + m^2 d^*|^2 \tag{1.236}$$
$$|\lambda^2 a + c|^2 \quad = |\lambda^2 a - a m^2 d^2/|d|^2|^2 = |a|^2 \cdot |\lambda^2 - m^2 d^2/|d|^2|^2 = |a|^2 \cdot |\lambda^2 - m^2 (d^*)^2/|d|^2|^2 \tag{1.237}$$
$$= |a|^2 |d|^{-4} \cdot |\lambda^2 |d|^2 - m^2 (d^*)^2|^2 \tag{1.238}$$
$$= |a|^2 |d|^{-4} \cdot |\lambda^2 d d^* - m^2 (d^*)^2|^2 \tag{1.239}$$
$$= |a|^2 |d|^{-4} |d^*|^2 \cdot |\lambda^2 d - m^2 d^*|^2 \tag{1.240}$$
$$\tag{1.241}$$

$$|\lambda^2 a - c|^2 \cdot |\lambda^2 a + c|^2 \quad = |a|^4 |d|^{-4} \cdot |\lambda^2 d + m^2 d^*|^2 \cdot |\lambda^2 d - m^2 d^*|^2 \tag{1.242}$$
$$= |a|^4 |d|^{-4} \cdot |(\lambda^2 d + m^2 d^*)(\lambda^2 d - m^2 d^*)|^2 \tag{1.243}$$
$$= |a|^4 |d|^{-4} \cdot |(\lambda^4 d^2 - m^4 (d^*)^2)|^2 \tag{1.244}$$
$$\tag{1.245}$$

We can now rearrange the provisional unit direction formula here, and compare the form with the previous complex variable example result.

$$p(\lambda) \quad = \frac{(\lambda^4 d^2 - m^4 (d^*)^2)}{|a|^4 |d|^{-4} \cdot |(\lambda^4 d^2 - m^4 (d^*)^2)|^2} \cdot \begin{pmatrix} \lambda^5 \cdot [\ 1/|d|^4 \cdot (d^*)^2 \cdot a^* a a^*\ ] \cdot (cb^* - \lambda^2 b a^*) a \\ +\lambda \cdot [\ m^2/|d|^2 \cdot a^* a a^*\ ] \cdot (cb^* - \lambda^2 b a^*) c \\ \hline (\lambda^4 |a|^2 - |c|^2) \\ \end{pmatrix} \tag{1.246}$$

$$p(\lambda) = \frac{\lambda \cdot (cb^* - \lambda^2 b a^*)}{(\lambda^4 |a|^2 - |c|^2)} \quad = \quad \frac{\lambda \cdot (cb^* - \lambda^2 b a^*)}{|a|^2 \cdot (\lambda^4 - m^4)} \qquad \leftarrow \text{ cf. when, } a, b, c \in \mathbb{C} \subset \mathbb{H}_R \tag{1.247}$$

Next, we bring in the, b, b^*, terms, where, $b = -(1/2) a m (sd + ds) d^*/|d|^2$, to evaluate the remaining factors at the roots,

$$(cb^* - \lambda^2 b a^*) \quad = ((-a m^2 d^2/|d|^2) \cdot (-(1/2) m/|d|^2 \cdot d(sd + ds)^* a^*) - \lambda^2 (-(1/2) m/|d|^2 \cdot a(sd + ds) d^*) \cdot a^*) \tag{1.248}$$
$$= (1/2) m/|d|^4 \cdot (\ m^2 \cdot a d^2 \cdot d(sd + ds)^* a^* + \lambda^2 |d|^2 \cdot a(sd + ds) d^* \cdot a^*\ ) \tag{1.249}$$

$$(cb^* - \lambda^2 b a^*) a \quad = (1/2) m/|d|^4 \cdot (\ m^2 \cdot a d^2 \cdot d(sd + ds)^* a^* + \lambda^2 |d|^2 \cdot a(sd + ds) d^* \cdot a^*\ ) \cdot a \tag{1.250}$$
$$= (1/2) m/|d|^4 \cdot (\ m^2 \cdot a d^2 \cdot d(sd + ds)^* a^* \cdot a + \lambda^2 |d|^2 \cdot a(sd + ds) d^* \cdot a^* \cdot a\ ) \tag{1.251}$$
$$= (1/2) m/|d|^4 \cdot a^* a a \cdot (\ m^2 \cdot d^2 \cdot d(sd + ds)^* + \lambda^2 |d|^2 \cdot (sd + ds) d^*\ ) \tag{1.252}$$

$$(cb^* - \lambda^2 b a^*) c \quad = (1/2) m/|d|^4 \cdot (\ m^2 \cdot a d^2 \cdot d(sd + ds)^* a^* + \lambda^2 |d|^2 \cdot a(sd + ds) d^* \cdot a^*\ ) \cdot (-a m^2 d^2/|d|^2) \tag{1.253}$$
$$= -(1/2) m^3/|d|^6 \cdot a^* a a \cdot (\ m^2 \cdot d^2 \cdot d(sd + ds)^* + \lambda^2 |d|^2 \cdot (sd + ds) d^*\ ) \cdot d^2 \tag{1.254}$$
$$\tag{1.255}$$

The lead parameter factors, a, can now all be combined into scalars, $a^*aa^* \cdot a^*aa = |a|^6$, and the last remaining denominator factor becomes, $(\lambda^4|a|^2 - |c|^2) = |a|^2 \cdot (\lambda^4 - m^4)$, since, $|c| = |-am^2d^2/|d|^2| = |a||m^2||d^2|/|d|^2 = |a|m^2$. So, both numerator and denominator then have overall factors of $|a|^6$, which enable us to cancel and remove them all from the entire formula, leaving us with just the roots (*We could have just set, $a = 1$, in the beginning, and avoided all this algebra tracking the $a's$ around, but it's useful to see it done this way at least once!*). Then, moving the $|d|^{-4}$ up from the denominator, pulling the $(1/2)$ factor all the way out to the front, and moving $m\lambda$ there, we have;

$$p(\lambda) = \frac{(\lambda^4 d^2 - m^4(d^*)^2)}{|(\lambda^4 d^2 - m^4(d^*)^2)|^2} \cdot \begin{pmatrix} \lambda^5 \cdot [\ 1/|d|^4 \cdot (d^*)^2\ ] \cdot a^*aa^* \cdot |d|^4 \cdot (cb^* - \lambda^2 ba^*)a \\ +\lambda \cdot [\ m^2/|d|^2\ ] \cdot a^*aa^* \cdot |d|^4 \cdot (cb^* - \lambda^2 ba^*)c \\ \hline |a|^6 \cdot (\lambda^4 - m^4) \end{pmatrix} \tag{1.256}$$

$$= \frac{(\lambda^4 d^2 - m^4(d^*)^2)}{|(\lambda^4 d^2 - m^4(d^*)^2)|^2} \cdot \begin{pmatrix} \lambda^5 \cdot [\ 1/|d|^4 \cdot (d^*)^2\ ] \cdot |a|^6 \cdot |d|^4 \cdot (1/2)m/|d|^4 \cdot (m^2d^3(sd+ds)^* + \lambda^2|d|^2(sd+ds)d^*) \\ +\lambda \cdot [\ m^2/|d|^2\ ] \cdot |a|^6 \cdot |d|^4 \cdot (-1/2)m^3/|d|^6 \cdot (m^2d^3(sd+ds)^* + \lambda^2|d|^2(sd+ds)d^*)d^2 \\ \hline |a|^6 \cdot (\lambda^4 - m^4) \end{pmatrix} \tag{1.257}$$

$$= \frac{m\lambda}{2} \cdot \frac{(\lambda^4 d^2 - m^4(d^*)^2)}{|(\lambda^4 d^2 - m^4(d^*)^2)|^2} \cdot \begin{pmatrix} [\ \lambda^4(d^*)^2\ ] \cdot (m^2d^3(sd+ds)^* + \lambda^2|d|^2(sd+ds)d^*) \\ -[\ m^4\ ] \cdot (m^2d^3(sd+ds)^* + \lambda^2|d|^2(sd+ds)d^*) \cdot d^2 \\ \hline |d|^4(\lambda^4 - m^4) \end{pmatrix} \tag{1.258}$$

Note here that, if we have, $a, c, b \in \mathbb{C} \subset \mathbb{H}_R$, then the two root terms, (s, d), would commute with each other. We'd then be able to move the d^2 factor from the right of the numerator, over to the left to merge with $[m^4]$, and so construct, $[\ \lambda^4(d^*)^2 - m^4d^2\ ]$, which is the "conjugate" of the outside numerator, $(\lambda^4 d^2 - m^4(d^*)^2)$, so these two would then combine to form the absolute value term, $|(\lambda^4 d^2 - m^4(d^*)^2|^2$, and cancel the matching term in the outside denominator, thus simplyfying the whole formula. In this situation, we'd then have,

$$p(\lambda) \quad = \frac{m\lambda}{2} \cdot \left(\frac{m^2d^3(sd+ds)^* + \lambda^2|d|^2(sd+ds)d^*}{|d|^4(\lambda^4 - m^4)} \right) = \frac{m\lambda}{1} \cdot \left(\frac{m^2d^2/|d|^2 \cdot s^* + \lambda^2 s}{(\lambda^4 - m^4)} \right) = \frac{m\lambda(u_1 + u_2)}{(\lambda^2 + m^2)} \tag{1.259}$$

Where we've used the identity, $d^2/|d|^2 = -u_1u_2$, in the numerator,[5] valid for this comlex number situation only. In this way, we see how the formula simplifies, in the complex variable case, to arrive at the result, $p(\lambda = m) = (u_1 + u_2)/2$, giving the average. The sub-expression, $(m^2d^3(sd+ds)^* + \lambda^2|d|^2(sd+ds)d^*)/|d|^4$, then, is responsible for producing the term, $(\lambda^2 - m^2)$, which cancels part of the denominator factor, $(\lambda^4 - m^4)$, turning it into a positive value, $(\lambda^2 + m^2)$, and thus removing the zero, which otherwise exists at the root, $\lambda = m$.

In the quaternion case, all the factors containing just the *difference of the roots*, d, d^*, and their powers, commute with each other, and it is only the presence of the *sum of the roots*, s, that prevents all these factors from commuting. What we'd like to do now, is move the d^2 factor, from the right in the numerator, over to the left to merge with m^4, so we can separate out the result with form like the complex variable example. To do this, we need to compute the commutator bracket, $[x, y] = xy - yx$, in two places, $[(sd + ds), d]$, and, $(sd + ds)^*, d]$. Let's see how these work out.

$$\begin{aligned} [\ (sd+ds), d\ ] &= (sd+ds)d - d(sd+ds) && (1.260)\\ &= 2(u_1^2 - u_2^2)(u_1 - u_2) - 2(u_1 - u_2)(u_1^2 - u_2^2) && (1.261)\\ &= 2(u_2du_1 - u_1du_2) && (1.262) \end{aligned}$$

$$\begin{aligned} [\ (sd+ds)^*, d\ ] &= (sd+ds)^*d - d(sd+ds)^* && (1.263)\\ &= 2(u_1^{*2} - u_2^{*2})(u_1 - u_2) - 2(u_1 - u_2)(u_1^{*2} - u_2^{*2}) && (1.264)\\ &= 2(u_1u_2^{*2} - u_1^{*2}u_2 + u_2u_1^{*2} - u_2^{*2}u_1) && (1.265) \end{aligned}$$

---

[5] $d^2/|d|^2 = (u_1 - u_2)^2/[(u_1 - u_2)(u_1^* - u_2^*)] = (u_1 - u_2)/(1/u_1 - 1/u_2) = (u_1 - u_2)u_1u_2/(u_2 - u_1) = -u_1u_2;\ \ u_1, u_2, d \in \mathbb{C} \subset \mathbb{H}_R$

We can now permute the factors using the relations;

$$(sd+ds)d = d(sd+ds) + [\,(sd+ds), d\,] \tag{1.266}$$
$$= d(sd+ds) + 2(u_2 du_1 - u_1 du_2) \tag{1.267}$$

$$(sd+ds)^* d = d(sd+ds)^* + [\,(sd+ds)^*, d\,] \tag{1.268}$$
$$= d(sd+ds)^* + 2(u_1 u_2^{*2} - u_1^{*2} u_2 + u_2 u_1^{*2} - u_2^{*2} u_1) \tag{1.269}$$

We have to use these relations *twice* in succession, since we're moving the square factor, d^2, over to the left.

$$(sd+ds)d^2 = d\{(sd+ds)d\} + [\,(sd+ds), d\,]d \tag{1.270}$$
$$= d\{d(sd+ds) + [\,(sd+ds), d\,]\} + [\,(sd+ds), d\,]d \tag{1.271}$$
$$= d^2(sd+ds) + d[\,(sd+ds), d\,] + [\,(sd+ds), d\,]d \tag{1.272}$$
$$= d^2(sd+ds) + \{d\,,\,[\,(sd+ds), d\,]\} \tag{1.273}$$

$$(sd+ds)^* d^2 = d\{(sd+ds)^* d\} + [\,(sd+ds)^*, d\,]d \tag{1.274}$$
$$= d\{d(sd+ds)^* + [\,(sd+ds)^*, d\,]\} + [\,(sd+ds)^*, d\,]d \tag{1.275}$$
$$= d^2(sd+ds)^* + d[\,(sd+ds)^*, d\,] + [\,(sd+ds)^*, d\,]d \tag{1.276}$$
$$= d^2(sd+ds)^* + \{d\,,\,[\,(sd+ds)^*, d\,]\} \tag{1.277}$$

where we've used the usual *anti-commutator*, $\{x\,,\,y\} = xy + yx$, to abbreviate the expression in the last step; not to be confused with the ordinary use of the curly brackets, { }, in the opening lines. The presence of the comma, { , }, distinguishes the anti-commutator, as usual, from the other usage of these brackets. We can then write, $p(\lambda)$,

$$p(\lambda) = \frac{m\lambda}{2} \cdot \frac{(\lambda^4 d^2 - m^4 (d^*)^2)}{|(\lambda^4 d^2 - m^4 (d^*)^2)|^2} \cdot \left(\begin{array}{c} [\,\lambda^4 (d^*)^2 - m^4 d^2\,] \cdot (m^2 d^3 (sd+ds)^* + \lambda^2 |d|^2 (sd+ds) d^*) \\ -[\,m^4\,] \cdot (m^2 d^3 \{d\,,\,[\,(sd+ds)^*, d\,]\} + \lambda^2 |d|^2 \{d\,,\,[\,(sd+ds), d\,]\} d^*) \\ \hline |d|^4 (\lambda^4 - m^4) \end{array} \right) \tag{1.278}$$

This then separates the provisional unit direction function into two parts,

$$p(\lambda) = \frac{m\lambda}{2} \cdot \left(\frac{m^2 d^3 (sd+ds)^* + \lambda^2 |d|^2 (sd+ds) d^*}{|d|^4 (\lambda^4 - m^4)} \right) \tag{1.279}$$
$$+ \frac{m\lambda}{2} \cdot \frac{(\lambda^4 d^2 - m^4 (d^*)^2)}{|(\lambda^4 d^2 - m^4 (d^*)^2)|^2} \cdot \left(\frac{-[\,m^4\,] \cdot (m^2 d^3 \{d\,,\,[\,(sd+ds)^*, d\,]\} + \lambda^2 |d|^2 \{d\,,\,[\,(sd+ds), d\,]\} d^*)}{|d|^4 (\lambda^4 - m^4)} \right) \tag{1.280}$$

The first term yields the familiar complex variable result, while the second term contains an additional contribution from the non-abelian algebra. When the variables commute, $[s, d] = 0$, and the commutator brackets all produce vanishishing terms, $[\,(sd+ds), d\,] = 0$, and, $[\,(sd+ds)^*, d\,] = 0$, then the second term drops out, leaving just the first term. In general, however, $[s, d] \neq 0$, and we have to consider the impact of the second term. But also note that even though the first term yields the complex variable result by itself, it still has embedded within it some remaining non-abelian character in the form of the *anti-commutator* expression, $\{s, d\} = sd + ds$, and it's conjugate. So, next we remove this remaining non-abelian residual, to get at the pure abelian result hidden within the formula.

$$[d, s] = ds - sd\,, \qquad \therefore \quad ds = sd + [d, s] \tag{1.281}$$

$$[\,m^2 d^3 (sd+ds)^* + \lambda^2 |d|^2 (sd+ds) d^*\,]/|d|^4 \tag{1.282}$$

$$= [\,m^2 d^3 (2sd + [d, s])^* + \lambda^2 |d|^2 (2sd + [d, s]) d^*\,]/|d|^4 \tag{1.283}$$
$$= 2(m^2 d^3 d^* s^* + \lambda^2 |d|^2 s d d^*)/|d|^4 + (m^2 d^3 [d, s]^* + \lambda^2 |d|^2 [d, s] d^*)/|d|^4 \tag{1.284}$$
$$= 2(m^2 |d|^2 d^2 s^* + \lambda^2 |d|^4 s)/|d|^4 + (m^2 d^3 [d, s]^* + \lambda^2 |d|^2 [d, s] d^*)/|d|^4 \tag{1.285}$$

$$= 2(m^2 \cdot d^2/|d|^2 \cdot s^* + \lambda^2 \cdot s) + (m^2 d^3 [d, s]^* + \lambda^2 |d|^2 [d, s] d^*)/|d|^4 \tag{1.286}$$

$$s^2 - d^2 = (u_1 + u_2)^2 - (u_1 - u_2)^2 = 2(u_1u_2 + u_2u_1) \tag{1.287}$$
$$d^2 = s^2 - 2(u_1u_2 + u_2u_1) \tag{1.288}$$
$$d^2 s^* = s^2 s^* - 2(u_1u_2 + u_2u_1)(u_1^* + u_2^*) \tag{1.289}$$
$$= |s|^2 s - 2(u_1 + u_2 + u_1u_2u_1^* + u_2u_1u_2^*) \tag{1.290}$$

$$[u_1, u_2] = u_1u_2 - u_2u_1 \tag{1.291}$$
$$u_1u_2 = u_2u_1 + [u_1, u_2] \tag{1.292}$$
$$u_2u_1 = u_1u_2 - [u_1, u_2] \tag{1.293}$$

$$u_1u_2u_1^* = u_2u_1u_1^* + [u_1, u_2]u_1^* = u_2 + [u_1, u_2]u_1^* \tag{1.294}$$
$$u_2u_1u_2^* = u_1u_2u_2^* - [u_1, u_2]u_2^* = u_1 - [u_1, u_2]u_2^* \tag{1.295}$$

$$d^2 s^* = |s|^2 s - 2(s + s + [u_1, u_2]u_1^* - [u_1, u_2]u_2^*) \tag{1.296}$$
$$d^2 s^* = (|s|^2 - 4)s - 2[u_1, u_2]d^* \tag{1.297}$$

$$[\, m^2 d^3 (sd + ds)^* + \lambda^2 |d|^2 (sd + ds)d^* \,]/|d|^4 \tag{1.298}$$
$$= 2(m^2 \cdot d^2/|d|^2 \cdot s^* + \lambda^2 \cdot s) + (m^2 d^3 [d, s]^* + \lambda^2 |d|^2 [d, s]d^*)/|d|^4 \tag{1.299}$$

$$= 2(m^2 \cdot \frac{(|s|^2 - 4)}{|d|^2} \cdot s + \lambda^2 \cdot s) - 4m^2 \cdot \frac{[u_1, u_2]d^*}{|d|^2} + \frac{(m^2 d^3 [d, s]^* + \lambda^2 |d|^2 [d, s]d^*)}{|d|^4} \tag{1.300}$$

$$= 2\left(\lambda^2 - m^2 \cdot \frac{(4 - |s|^2)}{|d|^2}\right) \cdot s \;\; + \;\; \frac{(m^2 d^3 [d, s]^* + \lambda^2 |d|^2 [d, s]d^*) - 4m^2 |d|^2 [u_1, u_2]d^*}{|d|^4} \tag{1.301}$$

If we work out the relation between, $|s|^2$ and $|d|^2$, we get,

$$|s|^2 = (u_1 + u_2)(u_1^* + u_2^*) = 1 + 1 + u_2u_1^* + u_1u_2^* = 2 + u_1u_2^* + u_2u_1^* \tag{1.302}$$
$$|d|^2 = (u_1 - u_2)(u_1^* - u_2^*) = 1 + 1 - u_2u_1^* - u_1u_2^* = 2 - u_1u_2^* - u_1u_2^* \tag{1.303}$$
$$|s|^2 + |d|^2 = 4 \tag{1.304}$$
$$|d|^2 = 4 - |s|^2 \tag{1.305}$$
$$\frac{(4 - |s|^2)}{|d|^2} \equiv 1 \tag{1.306}$$
$$\tag{1.307}$$

So the first term in the provisional unit direction function separates into abelian and non-abelian parts.

$$p(\lambda) \;\; = \;\; \frac{m\lambda}{2} \cdot 2\left(\frac{\lambda^2 - m^2}{\lambda^4 - m^4}\right) \cdot s \;\; + \;\; \frac{m\lambda}{2} \cdot \frac{(m^2 d^3 [d, s]^* + \lambda^2 |d|^2 [d, s]d^*) - 4m^2 |d|^2 [u_1, u_2]d^*}{|d|^4(\lambda^4 - m^4)} \;\; + \cdots \tag{1.308}$$

Noting that[6], $[d, s] = 2[u_1, u_2]$, using the anti-commutator, $\{d, s\} = (ds + sd)$, and putting this altogether...

$$p(\lambda) \;\; = \left(\frac{m\lambda}{\lambda^2 + m^2}\right) \cdot s \;\; + \;\; \frac{m\lambda}{2} \cdot \frac{(m^2 d^3 [d, s]^* + \lambda^2 |d|^2 [d, s]d^*) - 2m^2 |d|^2 [d, s]d^*}{|d|^4(\lambda^4 - m^4)} \tag{1.309}$$

$$+ \frac{m\lambda}{2} \cdot \frac{(\lambda^4 d^2 - m^4 (d^*)^2)}{|(\lambda^4 d^2 - m^4 (d^*)^2)|^2} \cdot \left(\frac{-[\, m^4 \,] \cdot (m^2 d^3 \{d, [\{d, s\}^*, d\,]\} + \lambda^2 |d|^2 \{d, [\{d, s\}, d\,]\} d^*)}{|d|^4(\lambda^4 - m^4)}\right) \tag{1.310}$$

---

[6] Using, $[d, s] = ds - sd = (u_1 - u_2)(u_1 + u_2) - (u_1 + u_2)(u_1 - u_2) = 2(u_1u_2 - u_2u_1) = 2[u_1, u_2]$.

Now we need to observe some bracket identities. For any two quaternions, A, B, the commutator, $[A, B]$, is a vector, i.e., $S([A, B]) = 0$, the scalar part is zero. This means that conjugating the bracket just introduces an overall minus sign, $[A, B]^* = -[A, B]$. Then, the commutator of a quaternion and a scalar is zero, hence, $[A, B + B^*] = 0$, but the commutator is a linear operator, so, $[A, B + B^*] = [A, B] + [A, B^*]$, this means that we can also write, $[A, B^*] = -[A, B]$. We then have a general rule for conjugation and the commutator, allowing us to interchangeably write any of the following forms, $[A, B]^* = [A, B^*] = [A^*, B] = [B^*, A^*] = -[A^*, B^*] = -[A, B]$.

Similarly, the anti-commutator is a linear operator, thus, $\{A, B+B^*\} = \{A, B\}+\{A, B^*\}$, and even nested brackets retain linearity, thus, $\{A, [B+B^*, C]\} = \{A, [B, C]\}+\{A, [B^*, C]\}$, and so on. With this in mind, we note the following identity relations for the (d, s) variables;

$$\begin{aligned}
[d, s]^* &\equiv -[d, s] && (1.311)\\
[d, s^*] &\equiv -[d, s] && (1.312)\\
[d^*, s] &\equiv -[d, s] && (1.313)\\
\{d, [\{d, s\}^*, d\,]\} &\equiv -\{d, [\{d, s\}, d\,]\} && (1.314)\\
[d, s] \cdot d &\equiv d^* \cdot [d, s] && (1.315)\\
[d, s] \cdot d^* &\equiv d \cdot [d, s] && (1.316)\\
[\{d, s\}, d\,] &\equiv -\{[d, s], d\} && (1.317)\\
d \cdot \{d, [\{d, s\}^*, d\,]\} &\equiv -\{d, [\{d, s\}, d\,]\} \cdot d^* && (1.318)
\end{aligned}$$

For example, to prove, $[d, s] \cdot d^* \equiv d \cdot [d, s]$, we make use of the conjugation and commutator property,

$$\begin{aligned}
d[d, s] &= -d[d, s]^* = -d(ds - sd)^* = -d(s^*d^* - d^*s^*) = -(ds^*d^* - dd^*s^*) = -(ds^*d^* - s^*dd^*) && (1.319)\\
&= -(ds^* - s^*d)d^* = -[d, s^*]d^* = -(-[d, s])d^* = [d, s]d^* && (1.320)
\end{aligned}$$

So now, in the expression for, $p(\lambda)$, we have three terms. The first is the abelian term like complex variables, and then there are two non-abelian term contributions. Consider the first non-abelian term.

$$\begin{aligned}
&+ \frac{m\lambda}{2} \cdot \frac{(m^2d^3 \cdot [d, s]^* + \lambda^2|d|^2 \cdot [d, s]d^*) - 2m^2|d|^2 \cdot [d, s]d^*}{|d|^4(\lambda^4 - m^4)}\\
&= + \frac{m\lambda}{2} \cdot \frac{(-m^2d^3 \cdot [d, s] + \lambda^2|d|^2 \cdot d[d, s]) - 2m^2|d|^2 \cdot d[d, s]}{|d|^4(\lambda^4 - m^4)}\\
&= + \frac{m\lambda}{2} \cdot \frac{(\lambda^2 - m^2)d^* - m^2(d + d^*)}{|d|^4(\lambda^4 - m^4)} \cdot dd \cdot [d, s] && (1.321)
\end{aligned}$$

Next, consider just the factor in the rightmost parenthesis for the second non-abelian term,

$$\cdot \left(\frac{-[\,m^4\,] \cdot \left(m^2d^2 \cdot d\{d, [\{d, s\}^*, d]\} + \lambda^2|d|^2 \cdot \{d, [\{d, s\}, d]\}d^*\right)}{|d|^4(\lambda^4 - m^4)} \right) \tag{1.322}$$

$$= \cdot \left(\frac{-[\,m^4\,] \cdot \left(-m^2d^2 \cdot \{d, [\{d, s\}, d]\}d^* + \lambda^2|d|^2 \cdot \{d, [\{d, s\}, d]\}d^*\right)}{|d|^4(\lambda^4 - m^4)} \right) \tag{1.323}$$

$$= \cdot \left(\frac{-[\,m^4\,] \cdot \left(-m^2d + \lambda^2d^*\right)}{|d|^4(\lambda^4 - m^4)} \right) \cdot d \cdot \{d, [\{d, s\}, d]\}d^* \tag{1.324}$$

Now lets look at the very last factor on the right of this,

$$\begin{aligned}
[\{d, s\}, d\,] &= -\{[d, s], d\} = -[d, s]d - d[d, s] = -d^*[d, s] - d[d, s] = -(d + d^*) \cdot [d, s] && (1.325)\\
\{d, [\{d, s\}, d\,]\} &= -d(d + d^*)[d, s] - (d + d^*) \cdot [d, s]d = -d(d + d^*)[d, s] - (d + d^*) \cdot d^*[d, s] && (1.326)\\
&= -(dd + dd^* + dd^* + d^*d^*) \cdot [d, s] && (1.327)\\
&= -(d + d^*)^2 \cdot [d, s] && (1.328)\\
\{d, [\{d, s\}, d\,]\} \cdot d^* &= -(d + d^*)^2 \cdot [d, s]d^* = -(d + d^*)^2 \cdot d[d, s] && (1.329)\\
d \cdot \{d, [\{d, s\}, d\,]\} \cdot d^* &= -d \cdot (d + d^*)^2 \cdot d[d, s] = -(d + d^*)^2 \cdot dd \cdot [d, s] && (1.330)
\end{aligned}$$

So now, we can re-write the provisional unit direction function, $p(\lambda)$, in terms of one commutator, $[d,s]$, which encapsulates all the non-abelian contribution,

$$p(\lambda) \quad = \left(\frac{m\lambda}{\lambda^2+m^2}\right)\cdot s \quad + \quad \frac{m\lambda}{2}\cdot\frac{(\lambda^2-m^2)d^*-m^2(d+d^*)}{|d|^4(\lambda^4-m^4)}\cdot dd\cdot[d,s] \tag{1.331}$$

$$+\frac{m\lambda}{2}\cdot\frac{(\lambda^4d^2-m^4(d^*)^2)}{|(\lambda^4d^2-m^4(d^*)^2)|^2}\cdot\left(\frac{+[\,m^4\,]\cdot\left(-m^2d+\lambda^2d^*\right)\cdot(d+d^*)^2}{|d|^4(\lambda^4-m^4)}\right)\cdot dd\cdot[d,s] \tag{1.332}$$

Splitting up the first non-abelian term, putting part with, $-m^2(d+d^*)$, back into the non-abelian second term;

$$p(\lambda) = \left(\frac{m\lambda}{\lambda^2+m^2}\right)\cdot s \quad + \quad \frac{m\lambda}{(\lambda^2+m^2)}\cdot\frac{d\cdot[d,s]}{2|d|^2}$$
$$+\frac{m\lambda}{2}\cdot\frac{(\lambda^4d^2-m^4(d^*)^2)}{|(\lambda^4d^2-m^4(d^*)^2)|^2}\cdot\left(\begin{array}{c}-m^2\cdot[\,\lambda^4(d^*)^2-m^4d^2\,]\\ \dfrac{+[\,m^4\,]\cdot\left(-m^2d+\lambda^2d^*\right)(d+d^*)}{|d|^4(\lambda^4-m^4)}\end{array}\right)\cdot(d+d^*)\cdot dd\cdot[d,s] \tag{1.333}$$

The numerator within the big parenthesis then reduces to, $(\lambda^2-m^2)\cdot m^2\cdot[m^2d-\lambda^2d^*]\cdot d^*$, providing us with that critical factor, (λ^2-m^2), that removes the zero from the denominator, turning the, (λ^4-m^4), into a more manegeable, (λ^2+m^2), and the rightmost, d^*, in this resultant numerator, then combines with one, d, from the, dd, outside factor, to produce, $|d|^2$, then reducing the denominator's, $|d|^4$, to just, $|d|^2$, we can then rearrange the formula for, $p(\lambda)$, as,

$$p(\lambda) = \left(\frac{m\lambda}{\lambda^2+m^2}\right)\cdot s \quad + \quad \frac{m\lambda}{(\lambda^2+m^2)}\cdot\frac{d\cdot[d,s]}{2|d|^2}$$
$$+\frac{m\lambda}{2}\cdot\frac{(\lambda^4d^2-m^4(d^*)^2)}{|(\lambda^4d^2-m^4(d^*)^2)|^2}\cdot\left(\frac{m^2\cdot(m^2d-\lambda^2d^*)}{|d|^2(\lambda^2+m^2)}\right)\cdot(d+d^*)\cdot d\cdot[d,s] \tag{1.334}$$

re-organizing terms,

$$\begin{aligned} p(\lambda) \quad &= \quad \left(\frac{m\lambda}{\lambda^2+m^2}\right)\cdot s \\ &+ \quad \left(\frac{m\lambda}{\lambda^2+m^2}\right)\cdot\left[\,1 \quad - \quad \left(\frac{(\lambda^4d^2-m^4(d^*)^2)(\lambda^2m^2d^*-m^4d)}{|(\lambda^4d^2-m^4(d^*)^2)|^2}\right)\cdot(d^*+d)\right]\cdot\frac{d\cdot[d,s]}{2|d|^2} \end{aligned} \tag{1.335}$$

When, $\lambda=m$, and, $d^2\neq(d^*)^2$, this formula reduces to,

$$p(\lambda) \quad = \quad \left(\frac{1}{2}\right)\cdot s \quad + \quad \left(\frac{1}{2}\right)\cdot\left[\,1 \quad - \quad \left(\frac{(d^2-(d^*)^2)(d^*-d)}{|(d^2-(d^*)^2)|^2}\right)\cdot(d^*+d)\right]\cdot\frac{d\cdot[d,s]}{2|d|^2} \tag{1.336}$$

$$= \quad \left(\frac{1}{2}\right)\cdot s \quad + \quad \left(\frac{1}{2}\right)\cdot\left[\,1 \quad - \quad \left(\frac{|(d^2-(d^*)^2)|^2}{|(d^2-(d^*)^2)|^2}\right)\right]\cdot\frac{d\cdot[d,s]}{2|d|^2} \tag{1.337}$$

$$= \quad \left(\frac{1}{2}\right)\cdot s \quad + \quad \left(\frac{1}{2}\right)\cdot[\,1 \quad - \quad 1\,]\cdot\frac{d\cdot[d,s]}{2|d|^2} \tag{1.338}$$

$$= \quad \left(\frac{1}{2}\right)\cdot s \tag{1.339}$$

Because the conjugate commutes with its own variable, $[d, d^*] = 0$, we have, $(d^* - d)(d^* + d) = ((d^*)^2 - d^2)$, the usual *difference of squares* applies, which otherwise dosn't usually apply to quaternion variables. So, we've established that for quaternions, $p(\lambda = m) = (u_1 + u_2)/2$, for this case with "*same magnitudes, different directions*," the same as for complex variables, and we no longer have to *assume* that setting, $\lambda = m$, in the formula for, $p(\lambda)$, gives us *the average* of the true unit directions. We have the proof. And our previous exploratory calculations are now justified.

But, what happens when, $d^2 = (d^*)^2$? We just excluded this, seemingly arbitrarily, to establish our result. However, we must include all cases. So, let's see. When these parameters are equal, the numerator term, $(\lambda^4 d^2 - m^4 (d^*)^2) = d^2(\lambda^4 - m^4)$, helps to cancel part of the denominator, $|(\lambda^4 d^2 - m^4 (d^*)^2)|^2 = |d|^4 \cdot (\lambda^4 - m^4)^2$, which is nice. But, we're still left with a remaining factor, $(\lambda^4 - m^4)$, in the denominator.

$$\begin{aligned} p(\lambda) &= \left(\frac{m\lambda}{\lambda^2 + m^2}\right) \cdot s \\ &+ \left(\frac{m\lambda}{\lambda^2 + m^2}\right) \cdot \left[1 - \left(\frac{d^2(\lambda^2 m^2 d^* - m^4 d)}{|d|^4(\lambda^4 - m^4)}\right) \cdot (d^* + d)\right] \cdot \frac{d \cdot [d, s]}{2|d|^2} \end{aligned} \tag{1.340}$$

If, $d^2 = (d^*)^2$, then, $d^2 \in \mathbb{R}$, and we have three situations; $d^2 < 0,\ d^2 = 0,\ d^2 > 0$. If, $d^2 > 0$, then, $d \in \mathbb{R}$, and, $d^* = d,\ |d|^2 = d^2$,

$$d^2 \geq 0,\ d \in \mathbb{R}: \tag{1.341}$$

$$p(\lambda) = \left(\frac{m\lambda}{\lambda^2 + m^2}\right) \cdot s + \left(\frac{m\lambda}{\lambda^2 + m^2}\right) \cdot \left[1 - \left(\frac{(\lambda^2 m^2 - m^4)}{(\lambda^4 - m^4)}\right) \cdot (2)\right] \cdot \frac{d \cdot [d, s]}{2|d|^2} \tag{1.342}$$

So, the parameter, d, drops out of the term inside the large square brackets. But, whatever the value inside these square brackets, since d is now a scalar, the commutator outside vanishes identically, $[d, s] \equiv 0$, so the whole non-abelian part drops out of the formula, leaving just the first term, $p = (m\lambda s/(\lambda^2 + m^2)$, so the solution method still finds the average of the units in this case, by setting, $\lambda = m$. To evaluate, $d = 0$, consider what happens when we start with, $d \in \mathbb{R},\ |d| > 0$, and then reduce the magnitude, $|d| \to 0$. Then, the inside term consists of a ratio of two expressions with same order in $|d|$, $\mathcal{O}(|d|^4)/\mathcal{O}(|d|^4)$, so remains bounded, while outside the commutator is identically zero again, $[d, s] \equiv 0$, because we're commuting a real scalar, thus we have the same result as $d^2 > 0$. However, $d = 0$, also means, $u_1 - u_2 = 0$, so that the direction units are the same, $u_1 = u_2 = u$, we have the completely degenerate situation of *same magnitudes, same directions*, which is easily solved by the two-step method. Here, $p(\lambda = m) = 1/2 \cdot (u + u) = u$, and thus, p, is a true unit direction, $|p| = 1$, and the usual constraint used to find λ is easily satisfied.

If, $d^2 < 0$, then, $d \in \mathbb{H},\ d \notin \mathbb{R},\ S(d) = 0$, and so, d, is a pure quaternion, i.e. a vector, thus, $d^* = -d$, hence, $(d^* + d) = (-d + d) \equiv 0$. The formula for p reduces to,

$$d^2 < 0,\ d \in \mathbb{H} - \mathbb{R}: \tag{1.343}$$

$$p(\lambda) = \left(\frac{m\lambda}{\lambda^2 + m^2}\right) \cdot s + \left(\frac{m\lambda}{\lambda^2 + m^2}\right) \cdot \frac{d \cdot [d, s]}{2|d|^2} \tag{1.344}$$

When, $\lambda = m$, we have,

$$d^2 < 0,\ d \in \mathbb{H} - \mathbb{R}: \tag{1.345}$$

$$p(\lambda = m) = \frac{1}{2} \cdot s + \frac{1}{2} \cdot \frac{d \cdot [d, s]}{2|d|^2}$$

$$\text{but,}\quad b = -1/2 \cdot am(sd + ds)d^*/|d|^2 = -1/2 \cdot am(2sd + [d, s])d^*/|d|^2 \tag{1.346}$$

$$= -1/2 \cdot am(2s|d|^2 + \underline{[d, s] \cdot d^*})/|d|^2 = -1/2 \cdot am(2s|d|^2 + \underline{d \cdot [d, s]})/|d|^2 \tag{1.347}$$

$$= -1/2 \cdot am(2s + d[d, s]/|d|^2) \tag{1.348}$$

$$-a^{-1}b/m = s + 1/2 \cdot d[d, s]/|d|^2 \tag{1.349}$$

$$\therefore\quad p(\lambda = m) = -1/2 \cdot a^{-1}b/m \tag{1.350}$$

$$c = -am^2d^2/|d|^2\ , \qquad |c| = |a|m^2|d^2|/|d|^2 = |a|m^2\ , \qquad m = \sqrt{|c|/|a|} \tag{1.351}$$

$$p(\lambda = m) \quad = \quad -1/2 \cdot \frac{a^*b}{|a|} \cdot \sqrt{\frac{|a|}{|c|}} \quad = \quad -1/2 \cdot \frac{a^*b}{\sqrt{|ac|}} \quad \overset{?}{\neq} \quad \frac{1}{2} \cdot s \tag{1.352}$$

So now we have a simple expression for $p(\lambda = m)$, and it does "*not seem*" to be *the average of the roots* ! What happened? Everything seemed to be going so well. In every other situation, but this one exceptional condition, $d^2 \in \mathbb{R},\ d^2 < 0$, the evaluation of the provisional direction unit, at the critical point, $\lambda = m = \sqrt{|c|/|a|}$, produces the *average* of the true units of the roots, $p(\lambda = m) = (u_1 + u_2)/2$. But, in this one special case, the principle seems violated. It has an extra term, drifting away from the average, by a somewhat quantum mechanical looking commutator expression, reminding us of the *Heisenberg uncertainty principle* that limits the precision with which we can know a value when attempting to measure a result in those cases where the operators don't commute.

$$p = \frac{(u_1 + u_2)}{2} \quad + \quad \frac{u_1 - u_2}{2|u_1 - u_2|^2} \cdot [u_1, u_2] \tag{1.353}$$

Notice that, when, $d^2 \in \mathbb{R}$, then, $a^*c \in \mathbb{R}$, and, $a^*c = -|a|^2m^2d^2/|d|^2 = -|a|^2m^2 \cdot \texttt{sign}[d^2]$, so that,

$$d^2 > 0 \ \texttt{.iff.}\ a^*c < 0, \quad \text{and,} \quad d^2 < 0 \ \texttt{.iff.}\ a^*c > 0.$$

Hence, our exceptional case here corresponds to the condition, $a^*c > 0$, which we've met before, in connection with the "infinity" roots case! That was also an exceptional case. So, perhaps, things aren't so bad after all. Let's take a closer look at this condition, $d^2 < 0$. We have two distinct situations when the difference of the true unit directions is a pure quaternion; i.e., $S(u_1 - u_2) = 0$. Either, $S(u_1) = S(u_2) = 0$, or, $S(u_1) = S(u_2) \neq 0$.

`Case I:` $u_1 \neq u_2,\ S(u_1) = S(u_2) = 0$. In this case, $sd + ds = 2(u_1^2 - u_2^2) = 2(-|u_1|^2 - (-|u_2|^2)) = 2(|u_2|^2 - |u_1|^2)$. But, $|u_1| = |u_2| = 1$, since these are "unit" quaternions. Thus, $sd + ds = 0$, and, $b = -(1/2)am(sd + ds)d^*/|d|^2 = 0$. So, the quadratic equation is of the form, $aq^2 + c = 0$. This is the very, $b = 0$, case, which we've excluded from the *two-hand two-step* method at the start. We already required, $b \neq 0$, in determining, $p(\lambda)$. And since, $a^*c > 0$, here, this is exactly the same "infinity" roots exceptional case we've met before. So, the exceptions are the same. There's nothing new here; Except, perhaps, seeing how the infinity roots special exception appears within the formula for the provisional unit direction function, when analysing things. Since, $b = 0$, here, we also have, $p = 0$, and obtain a special exceptional identity relation between the average of "*any two*" of the "infinity" of the roots, and their commutator.

$$\frac{(u_1 + u_2)}{2} \quad = \quad -\frac{u_1 - u_2}{2|u_1 - u_2|^2} \cdot [u_1, u_2] \qquad \forall u_1, u_2 \ \in \mathbb{H}, \text{with, } u_1^2 = -1, u_2^2 = -1. \tag{1.354}$$

`Case II:` $S(u_1) = S(u_2) \neq 0$. In this case, the scalar parts don't vanish, but are the same value. So, let's set this value to, $S(u_1) = S(u_2) = u_{00}$, and let the vector parts be, $V(u_1) = v_1$, and, $V(u_2) = v_2$; so that, $u_1 = u_{00} + v_1$, and, $u_2 = u_{00} + v_2$. Then, $sd + ds = 4u_{00}(v_1 - v_2)$, and, $d = (v_1 - v_2)$, so that, $d^* = -(v_1 - v_2)$, and then,

$$b \quad = +(1/2)4am \cdot u_{00}(v_1 - v_2)^2/|v_1 - v_2|^2 = -2m \cdot u_{00} \cdot a \tag{1.355}$$

$$c \quad = -am^2d^2/|d|^2 = -am^2(v_1 - v_2)^2/|v_1 - v_2|^2 = +m^2 \cdot a \tag{1.356}$$

$$0 \quad = aq^2 + bq + c = a \cdot (q^2 - 2mu_{00} \cdot q + m^2) \tag{1.357}$$

$$q \quad = mu_{00} \pm m\sqrt{-1}\sqrt{1 - u_{00}^2} = m \cdot \left(u_{00} \pm n\sqrt{1 - u_{00}^2}\right)\ , \qquad \forall n \in \mathbb{H},\ n^2 = -1 \tag{1.358}$$

$$p(\lambda = m) \quad = u_{00} \tag{1.359}$$

With the implicated values for, (a, b, c), we see the quadratic equation converts to one with real valued coefficients, just by dividing throughout by the lead, a. We then have an infinity of roots. Another infinity! In the case above, the exception occurred when, $b = 0$, which is "out of scope" of our two-step method. But, here, $b \neq 0$, and we have an infinity exception "within scope." However, we've seen this one before too. In the `offplane` "magic factor" solution. Notice, that if we take two opposite roots, $u_1 = u_{00} + n\sqrt{1 - u_{00}^2}$, and, $u_2 = u_{00} - n\sqrt{1 - u_{00}^2}$, their *average* is just, $p = u_{00}$. This is how we looked at the problem before. But, now we've got some more insight into the infinite roots. Because of the special commutator expression term, showing up in, $p(\lambda)$, we don't actually need to pick special "opposite" units, to make our average, the way we reasoned before, guided by our intuition. Pick "any two" roots out of the infinite set, and plug them into the formula for, $p(\lambda)$, and like magic, out pops the very same "average" value, $p = u_{00}$. To confirm this corresponds exactly to the `offplane` case, we compute the residual related expression, $w = \lambda^2 a^*(a(\lambda p)^2 + b(\lambda p) + c)$, evaluating at our given values, (λ, p, a, b, c), here, where we now find, $w = m^4|a|^2(1 - u_{00}^2),\ \implies\ w \in \mathbb{R},\ w > 0$, since, $u_{00}^2 < 1$. Q.E.D.

`Case 2 - Same direction, different magnitudes:`
$q_1 = \lambda_1 \cdot u,\ q_2 = \lambda_2 \cdot u,\ \ \lambda_1, \lambda_2 \in \mathbb{R},\ \lambda_1, \lambda_2 > 0,\ \lambda_1 \neq \lambda_2;\ u \in \mathbb{H}_R,\ |u| = 1.$

Consider the following quadratic equation, with solutions, q_1 and q_2, and the respective '*reconstructed*' pair of equivalent equations;

$$aq^2 + bq + c = 0 \tag{1.360}$$

$$\underline{q^2 + a^{-1}bq + a^{-1}c = 0} \tag{1.361}$$

$$(q - q_1)(q - q_2) = 0 \tag{1.362}$$
$$(q - q_2)(q - q_1) = 0 \tag{1.363}$$

$$q^2 - q_1 \cdot q - q \cdot q_2 + q_1 q_2 = 0 \tag{1.364}$$
$$q^2 - q_2 \cdot q - q \cdot q_1 + q_2 q_1 = 0 \tag{1.365}$$

$$q^2 - \lambda_1 u \cdot q - q \cdot \lambda_2 u + \lambda_1 u \lambda_2 u = 0 \tag{1.366}$$
$$q^2 - \lambda_2 u \cdot q - q \cdot \lambda_1 u + \lambda_2 u \lambda_1 u = 0 \tag{1.367}$$

$$\lambda_1 q^2 - \lambda_1^2 u \cdot q - \lambda_1 q \cdot \lambda_2 u + \lambda_1^2 u \lambda_2 u = 0 \tag{1.368}$$
$$\lambda_2 q^2 - \lambda_2^2 u \cdot q - \lambda_2 q \cdot \lambda_1 u + \lambda_2^2 u \lambda_1 u = 0 \tag{1.369}$$

$$(\lambda_1 - \lambda_2) \cdot q^2 - (\lambda_1^2 - \lambda_2^2) u \cdot q + \lambda_1 \lambda_2 (\lambda_1 - \lambda_2) u^2 = 0 \tag{1.370}$$

$$\underline{q^2 - (\lambda_1 + \lambda_2) uq + \lambda_1 \lambda_2 u^2 = 0} \tag{1.371}$$

$$\therefore \quad a^{-1}b = -(\lambda_1 + \lambda_2)u \tag{1.372}$$
$$a^{-1}c = \lambda_1 \lambda_2 u^2 \tag{1.373}$$

From analysing the reconstructions, as shown, we establish the relationship between the original quaternion equation parameters, a, b, c, and the solution roots, $q_1 = \lambda_1 u$, and, $q_2 = \lambda_2 u$. Then, consider, now, the formula under test, our usual "*provisional*" unit direction function, p,

$$p(\lambda) = \frac{\lambda^5(\lambda^4 a^* a a^* - c^* a c^*)(cb^* - \lambda^2 b a^*)a + \lambda(c^* c c^* - \lambda^4 a^* c a^*)(cb^* - \lambda^2 b a^*)c}{(\lambda^4 |a|^2 - |c|^2) \cdot |\lambda^2 a - c|^2 \cdot |\lambda^2 a + c|^2} \tag{1.374}$$

Let, $\lambda_1 = m, \lambda_2 = n$, and without any loss of generality, we set, $a = 1$, so that, $b = a^{-1}b = -(m+n)u$, and the other, $c = a^{-1}c = mnu^2$. Then, substituting these values of the parameters, (a, b, c), into, $p(\lambda)$, we have,

$$p(\lambda) = \left(\begin{array}{c} \lambda^5(\lambda^4 - m^2 n^2 (u^*)^4)(-mnu^2(m+n)u^* + \lambda^2(m+n)u) \\ +\lambda(m^3 n^3 (u^*)^2 - \lambda^4 mnu^2)(-mnu^2(m+n)u^* + \lambda^2(m+n)u)mnu^2 \\ \hline (\lambda^4 - |mn|^2) \cdot |\lambda^2 - mnu^2|^2 \cdot |\lambda^2 + mnu^2|^2 \end{array} \right) \tag{1.375}$$

All the powers of the units, u, u^*, commute with each other, and, $uu^* = 1$. So, we can simplify this formula.

$$p(\lambda) = \lambda \cdot \left(\frac{\lambda^4 \lambda^4 - \lambda^4 m^2 n^2 (u^*)^4 - \lambda^4 m^2 n^2 u^4 + m^4 n^4}{(\lambda^4 - m^2 n^2) \cdot |\lambda^2 - mnu^2|^2 \cdot |\lambda^2 + mnu^2|^2} \right) \cdot (-mn(m+n) + \lambda^2(m+n))u \tag{1.376}$$

We now insert the identity, $1 \equiv (u^*)^4 u^4$, into the numerator term to enable factoring of this expression.

$$p(\lambda) = \lambda \cdot \left(\frac{\lambda^4 \lambda^4 - \lambda^4 m^2 n^2 (u^*)^4 - \lambda^4 m^2 n^2 u^4 + m^4 n^4 (u^*)^4 u^4}{(\lambda^4 - m^2 n^2) \cdot |\lambda^2 - mnu^2|^2 \cdot |\lambda^2 + mnu^2|^2} \right) \cdot (-mn(m+n) + \lambda^2(m+n))u \tag{1.377}$$

Then we can simplify,

$$p(\lambda) \quad = \lambda \cdot \left(\frac{(\lambda^4 - m^2n^2(u^*)^4)(\lambda^4 - m^2n^2u^4)}{(\lambda^4 - m^2n^2) \cdot |\lambda^2 - mnu^2|^2 \cdot |\lambda^2 + mnu^2|^2} \right) \cdot (-mn(m+n) + \lambda^2(m+n))u \tag{1.378}$$

$$= \lambda \cdot \left(\frac{(\lambda^2 - mn(u^*)^2)(\lambda^2 + mn(u^*)^2) \cdot (\lambda^2 - mnu^2)(\lambda^2 + mnu^2)}{(\lambda^4 - m^2n^2) \cdot |\lambda^2 - mnu^2|^2 \cdot |\lambda^2 + mnu^2|^2} \right) \cdot (\lambda^2 - mn)(m+n)u \tag{1.379}$$

$$= \lambda \cdot \left(\frac{(\lambda^2 - mn(u^*)^2)(\lambda^2 - mnu^2) \cdot (\lambda^2 + mn(u^*)^2)(\lambda^2 + mnu^2)}{(\lambda^4 - m^2n^2) \cdot |\lambda^2 - mnu^2|^2 \cdot |\lambda^2 + mnu^2|^2} \right) \cdot (\lambda^2 - mn)(m+n)u \tag{1.380}$$

$$= \lambda \cdot \left(\frac{|\lambda^2 - mnu^2|^2 \cdot |\lambda^2 + mnu^2|^2}{(\lambda^4 - m^2n^2) \cdot |\lambda^2 - mnu^2|^2 \cdot |\lambda^2 + mnu^2|^2} \right) \cdot (\lambda^2 - mn)(m+n)u \tag{1.381}$$

$$= \lambda \cdot \left(\frac{1}{(\lambda^4 - m^2n^2)} \right) \cdot (\lambda^2 - mn)(m+n)u \quad = \quad \left(\frac{\lambda(m+n)}{(\lambda^2 + mn)} \right) \cdot u \tag{1.382}$$

The solution method is trying to find λ such that the constraint, $|p(\lambda)| = 1$, is met. This means that, we must satisfy the following,

$$|\lambda(m+n)u| = |(\lambda^2 + mn)| \tag{1.383}$$

First, the method tries to match, $\lambda = m$, and satisfy the constraint. So, to see if this is always possible, we replace, λ,

$$\lambda = m, \tag{1.384}$$
$$|m(m+n)u| = |(m^2 + mn)| \tag{1.385}$$
$$|m| \cdot |m+n| \cdot |u| = |m| \cdot |m+n| \tag{1.386}$$

Match succeeds, since, $|u| = 1$, and the method is able to determine the direction unit, p,

$$p(\lambda = m) \quad = \quad \left(\frac{m(m+n)}{(m^2 + mn)} \right) \cdot u \quad = \quad u \tag{1.387}$$

Then the method tries to match, $\lambda = n$, and satisfy the constraint. So, to see if this is always possible, we replace, λ,

$$\lambda = n, \tag{1.388}$$
$$|n(m+n)u| = |(n^2 + mn)| \tag{1.389}$$
$$|n| \cdot |m+n| \cdot |u| = |n| \cdot |n+m| \tag{1.390}$$

Match succeeds, since, $|u| = 1$, and the method is able to determine the direction unit, p,

$$p(\lambda = n) \quad = \quad \left(\frac{n(m+n)}{(n^2 + mn)} \right) \cdot u \quad = \quad u \tag{1.391}$$

Hence, *in principle*, there's no problem with the method in finding the solutions to the quadratic equation for the case of *Same direction, different magnitudes.*

`Corrollary:` Since, $a^{-1}b = -(\lambda_1 + \lambda_2)u$, and, $a^{-1}c = \lambda_1\lambda_2u^2$, we have an expression for the *sum of the roots*, $q_1 + q_2 = \lambda_1 u + \lambda_2 u = -a^{-1}b$, and we have an expression for the *product of the roots*, $q_1q_2 = \lambda_1\lambda_2u^2 = a^{-1}c$, which we may also write, $aq_1q_2 = \lambda_1\lambda_2au^2 = c$, following the form of the square term, aqq, with the lead parameter, a, multiplying on the left.

`Case 3 - Different magnitudes, different directions:` ✓!
$q_1 = \lambda_1 \cdot u_1,\ q_2 = \lambda_2 \cdot u_1,\ \ \lambda_1, \lambda_2 \in \mathbb{R},\ \lambda_1, \lambda_2 > 0,\ \lambda_1 \neq \lambda_2;\ u_1, u_2 \in \mathbb{H}_R,\ |u_1| = 1,\ |u_2| = 1,\ u_1 \neq u_2.$

Given the two roots, q_1 and q_2, consider the two implied *direct equations*;

$$aq_1^2 + bq_1 + c = 0 \tag{1.392}$$
$$aq_2^2 + bq_2 + c = 0 \tag{1.393}$$

Let, $\lambda_1 = m, \lambda_2 = n$,

$$am^2u_1^2 + bmu_1 + c = 0 \tag{1.394}$$
$$an^2u_2^2 + bnu_2 + c = 0 \tag{1.395}$$
$$a(m^2u_1^2 - n^2u_2^2) + b(mu_1 - nu_2) = 0 \tag{1.396}$$
$$b(mu_1 - nu_2) = -a(m^2u_1^2 - n^2u_2^2) \tag{1.397}$$
$$\rightarrow \quad b = -a(m^2u_1^2 - n^2u_2^2)(mu_1^* - nu_2^*)/|mu_1 - nu_2|^2 \tag{1.398}$$

$$anm^2u_1 + bmn + cnu_1^* = 0 \tag{1.399}$$
$$amn^2u_2 + bnm + cmu_2^* = 0 \tag{1.400}$$
$$a(nm^2u_1 - mn^2u_2) + c(nu_1^* - mu_2^*) = 0 \tag{1.401}$$
$$c(nu_1^* - mu_2^*) = -a(nm^2u_1 - mn^2u_2) \tag{1.402}$$
$$c(nu_1^{-1} - mu_2^{-1}) = -amn(mu_1 - nu_2) \tag{1.403}$$
$$cu_1^{-1}(mu_1 - nu_2)u_2^{-1} = amn(mu_1 - nu_2) \tag{1.404}$$
$$\rightarrow \quad c = amn(mu_1 - nu_2)u_2(mu_1^* - nu_2^*)u_1/|mu_1 - nu_2|^2 \tag{1.405}$$
$$\tag{1.406}$$

With the above procedure, we establish the relationship between the quadratic equation's paramters, (a, b, c), and the two roots, $q_1 = m \cdot u_1$, and, $q_2 = n \cdot u_2$. Then, consider the formula under test, our usual provisional unit direction function, $p(\lambda)$,

$$p = \frac{\lambda^5(\lambda^4 a^* a a^* - c^* a c^*)(cb^* - \lambda^2 b a^*)a + \lambda(c^* c c^* - \lambda^4 a^* c a^*)(cb^* - \lambda^2 b a^*)c}{(\lambda^4|a|^2 - |c|^2) \cdot |\lambda^2 a - c|^2 \cdot |\lambda^2 a + c|^2} \tag{1.407}$$

Now, without any loss in generality, we can set, $a = 1$,

$$p = \frac{\lambda^5(\lambda^4 - c^* c^*)(cb^* - \lambda^2 b) + \lambda(c^* c c^* - \lambda^4 c)(cb^* - \lambda^2 b)c}{(\lambda^4 - |c|^2) \cdot |\lambda^2 - c|^2 \cdot |\lambda^2 + c|^2} \tag{1.408}$$

Then, using the commutators, $[b, c] = bc - cb$, and, $[b^*, c] = b^*c - cb^*$, we move the factor, c, from the right side to left side in the rightmost numerator term, $(cb^* - \lambda^2 b)c \rightarrow c(cb^* - \lambda^2 b)$, and recall that, $[b^*, c] = -[b, c]$.

$$(cb^* - \lambda^2 b)c \quad = (cb^*c - \lambda^2 bc) = c(cb^* + [b^*, c]) - \lambda^2(cb + [b, c]) \tag{1.409}$$
$$= c(cb^* - \lambda^2 b) - (c + \lambda^2)[b, c] \tag{1.410}$$
$$\tag{1.411}$$

This lets us re-arrange, $p(\lambda)$,

$$p \quad = \lambda \cdot \left(\frac{(\lambda^4 - cc)(\lambda^4 - c^* c^*)}{(\lambda^4 - |c|^2) \cdot |\lambda^2 - c|^2 \cdot |\lambda^2 + c|^2}\right) \cdot (cb^* - \lambda^2 b) \quad + \quad \left(\frac{\lambda c(\lambda^4 - c^* c^*)(\lambda^2 + c)[b, c]}{(\lambda^4 - |c|^2) \cdot |\lambda^2 - c|^2 \cdot |\lambda^2 + c|^2}\right) \tag{1.412}$$

$$= \lambda \cdot \left(\frac{cb^* - \lambda^2 b}{\lambda^4 - |c|^2}\right) \quad + \quad \left(\frac{\lambda c(\lambda^2 - c^*)[b, c]}{(\lambda^4 - |c|^2) \cdot |\lambda^2 - c|^2}\right) \quad = \quad \lambda \cdot (\lambda^2 - c^*) \cdot \left(\frac{(\lambda^2 - c) \cdot (cb^* - \lambda^2 b) - c[b^*, c]}{(\lambda^4 - |c|^2) \cdot |\lambda^2 - c|^2}\right)$$

$$= \lambda \cdot (\lambda^2 - c^*) \cdot \left(\frac{(\lambda^2 - c) \cdot (cb^* - \lambda^2 b) - c(b^*c - cb^*)}{(\lambda^4 - |c|^2) \cdot |\lambda^2 - c|^2}\right) \tag{1.413}$$

So, we've rearranged the formula into something more suitable for the current analysis. We start with,

$$p(\lambda) \quad = \lambda \cdot (\lambda^2 - c^*) \cdot \left(\frac{cb^* \cdot (\lambda^2 - c) - \lambda^2(\lambda^2 - c) \cdot b}{(\lambda^4 - |c|^2) \cdot |\lambda^2 - c|^2} \right) \tag{1.414}$$

The requirement, $|p(\lambda)| = 1$, then means,

$$|\lambda \cdot (\lambda^2 - c^*) \cdot (cb^* \cdot (\lambda^2 - c) - \lambda^2(\lambda^2 - c) \cdot b)| \quad = |\lambda^4 - |c|^2| \cdot |\lambda^2 - c|^2 \tag{1.415}$$
$$|\lambda| \cdot |(\lambda^2 - c)| \cdot |(cb^* \cdot (\lambda^2 - c) - \lambda^2(\lambda^2 - c) \cdot b)| \quad = |\lambda^4 - |c|^2| \cdot |\lambda^2 - c|^2 \tag{1.416}$$
$$|\lambda| \cdot |(cb^* \cdot (\lambda^2 - c) - \lambda^2(\lambda^2 - c) \cdot b)| \quad = |\lambda^4 - |c|^2| \cdot |\lambda^2 - c| \tag{1.417}$$

Let's introduce some helpful variables and formulae.

$$u_1 = u_{10} + u_{11}i + u_{12}j + u_{13}k = u_{10} + v_1 \tag{1.418}$$
$$u_2 = u_{20} + u_{21}i + u_{22}j + u_{23}k = u_{20} + v_2 \tag{1.419}$$

consider..

$$u_1 = u_{10} + v_1 \tag{1.420}$$
$$u_1 u_1^* = (u_{10} + v_1)(u_{10} - v_1) = u_{10}^2 - v_1^2 = 1 \tag{1.421}$$
$$\therefore \quad v_1^2 = u_{10}^2 - 1 \tag{1.422}$$
$$u_1^2 = (u_{10} + v_1)(u_{10} + v_1) = u_{10}^2 + 2u_{10}v_1 + v_1^2 \tag{1.423}$$
$$= u_{10}^2 + 2u_{10}v_1 + u_{10}^2 - 1 \tag{1.424}$$
$$= 2u_{10}^2 - 1 + 2u_{10}v_1 \tag{1.425}$$
$$= 2u_{10}^2 - 1 + 2u_{10}(u_1 - u_{10}) \tag{1.426}$$
$$= 2u_{10}^2 - 1 + 2u_{10}u_1 - 2u_{10}^2 \tag{1.427}$$
$$= -1 + 2u_{10}u_1 \tag{1.428}$$

$\therefore$

$$\underline{u_1^2 = -1 + 2u_{10}u_1} \quad = \quad -1 + (u_1 + u_1^*)u_1 \tag{1.429}$$
$$\underline{u_2^2 = -1 + 2u_{20}u_2} \quad = \quad -1 + (u_2 + u_2^*)u_2 \tag{1.430}$$

$$b = -(m^2u_1^2 - n^2u_2^2)(mu_1^* - nu_2^*)/|mu_1 - nu_2|^2 \tag{1.431}$$
$$c = mn(mu_1 - nu_2)u_2(mu_1^* - nu_2^*)u_1/|mu_1 - nu_2|^2 \tag{1.432}$$

$$|c| = mn \cdot |(mu_1 - nu_2)| \cdot |u_2| \cdot |(mu_1^* - nu_2^*)| \cdot |u_1|/|mu_1 - nu_2|^2 \quad = \quad mn \tag{1.433}$$
$$(\lambda^4 - |c|^2) = (\lambda^4 - m^2n^2) \tag{1.434}$$

$$(\lambda^2 - c) = \lambda^2 - mn(mu_1 - nu_2)u_2(mu_1^* - nu_2^*)u_1/|mu_1 - nu_2|^2 \tag{1.435}$$
$$= [\ \lambda^2(mu_1 - nu_2)(mu_1^* - nu_2^*) - mn(mu_1 - nu_2)u_2(mu_1^* - nu_2^*)u_1\ ]/|mu_1 - nu_2|^2 \tag{1.436}$$
$$= \frac{(mu_1 - nu_2)}{|mu_1 - nu_2|^2} \cdot [\ \lambda^2(mu_1^* - nu_2^*) - mnu_2(mu_1^* - nu_2^*)u_1\ ] \tag{1.437}$$

Now let's take a look at the expression in square brackets.

$$[\ \lambda^2(mu_1^* - nu_2^*) - mnu_2(mu_1^* - nu_2^*)u_1\ ] \tag{1.438}$$
$$= [\ \lambda^2(mu_1^* - nu_2^*) - mn(mu_2 - nu_1)\ ] \tag{1.439}$$
$$= [\ \lambda^2(m(u_{10} - v_1) - n(u_{20} - v_2)) - mn(m(u_{20} + v_2) - n(u_{10} + v_1))\ ] \tag{1.440}$$
$$= [\ \lambda^2(mu_{10} - mv_1 - nu_{20} + nv_2) - mn(mu_{20} + mv_2 - nu_{10} - nv_1)\ ] \tag{1.441}$$
$$= [\ m(\lambda^2 + n^2)u_{10} - n(\lambda^2 + m^2)u_{20} - m(\lambda^2 - n^2)v_1 + n(\lambda^2 - m^2)v_2\ ] \tag{1.442}$$
$$= [\ m(\lambda^2 + n^2)u_{10} - n(\lambda^2 + m^2)u_{20} - m(\lambda^2 - n^2)(u_1 - u_{10}) + n(\lambda^2 - m^2)(u_2 - u_{20})\ ] \tag{1.443}$$
$$= [\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2\ ] \tag{1.444}$$

So that,

$$(\lambda^2 - c) \quad = \frac{(mu_1 - nu_2)}{|mu_1 - nu_2|^2} \cdot [\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2\ ] \tag{1.445}$$

So, we have estabished the formula in the constraint on the R.H.S, in terms of the two roots, $q_1 = mu_1, q_2 = nu_2$.

$$|\lambda^4 - |c|^2| \cdot |\lambda^2 - c| \quad = \frac{|\lambda^4 - m^2n^2|}{|mu_1 - nu_2|} \cdot |\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2\ | \tag{1.446}$$

Now we look at the L.H.S. of this constraint:

$$|\lambda| \cdot |(cb^* \cdot (\lambda^2 - c) - \lambda^2(\lambda^2 - c) \cdot b)| \quad = \quad |\lambda^4 - |c|^2| \cdot |\lambda^2 - c| \tag{1.447}$$

Using,

$$c \quad = \frac{(mu_1 - nu_2)}{|mu_1 - nu_2|^2} \cdot mnu_2(mu_1^* - nu_2^*)u_1 \tag{1.448}$$

$$(\lambda^2 - c) \quad = \frac{(mu_1 - nu_2)}{|mu_1 - nu_2|^2} \cdot [\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2\ ] \tag{1.449}$$

we replace these, leaving in place, b, b^*, in the expression, and then introduce the abbreviations, $s = mu_1 + nu_2$, and, $d = mu_1 - nu_2$, for the *sum* and the *difference* of the roots, to reduce clutter in these formulas.

$$(cb^*(\lambda^2 - c) - \lambda^2(\lambda^2 - c)b) \tag{1.450}$$

$$= \frac{(mu_1 - nu_2)}{|mu_1 - nu_2|^2} \cdot \begin{bmatrix} mnu_2(mu_1^* - nu_2^*)u_1 \cdot b^*(\lambda^2 - c) \\ -\lambda^2 \cdot [\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2\ ] \cdot b \end{bmatrix} \tag{1.451}$$

$$= \frac{(mu_1 - nu_2)}{|mu_1 - nu_2|^2} \cdot \begin{bmatrix} mnu_2(mu_1^* - nu_2^*)u_1 \cdot b^* \dfrac{(mu_1 - nu_2)}{|mu_1 - nu_2|^2} \cdot [\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2\ ] \\ -\lambda^2 \cdot [\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2\ ] \cdot b \end{bmatrix} \tag{1.452}$$

$$= \frac{(mu_1 - nu_2)}{|mu_1 - nu_2|^4} \cdot \begin{bmatrix} mnu_2(mu_1^* - nu_2^*)u_1 \cdot b^* \dfrac{(mu_1 - nu_2)}{1} \cdot [\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2\ ] \\ -|mu_1 - nu_2|^2 \cdot \lambda^2 \cdot [\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2\ ] \cdot b \end{bmatrix} \tag{1.453}$$

$$= \frac{d}{|d|^4} \cdot \begin{bmatrix} mnu_2d^*u_1 \cdot b^*d \cdot [\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2\ ] \\ -|d|^2 \cdot \lambda^2 \cdot [\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2\ ] \cdot b \end{bmatrix} \tag{1.454}$$

Good...now, the objective is to move that b parameter, from the right, all the way over to the left of the inner square bracketed expression. To do this, we need two commutators, $[\ u_1, b\ ]$, and, $[\ u_2, b\ ]$. We get,

$$= \frac{d}{|d|^4} \cdot \begin{bmatrix} mnu_2d^*u_1 \cdot b^*d \cdot [\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2\ ] \\ -|d|^2 \cdot \lambda^2 \cdot b \cdot [\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2\ ] \\ -|d|^2 \cdot \lambda^2 \cdot [\ -m(\lambda^2 - n^2) \cdot [\ u_1, b\ ] + n(\lambda^2 - m^2) \cdot [\ u_2, b\ ]\ ] \end{bmatrix} \tag{1.455}$$

which, then becomes,

$$= \frac{d}{|d|^4} \cdot \begin{bmatrix} (mnu_2d^*u_1 \cdot b^*d - |d|^2 \cdot \lambda^2 \cdot b) \cdot [\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2\ ] \\ -|d|^2 \cdot \lambda^2 \cdot [\ -m(\lambda^2 - n^2) \cdot [\ u_1, b\ ] + n(\lambda^2 - m^2) \cdot [\ u_2, b\ ]\ ] \end{bmatrix} \tag{1.456}$$

The next task, is to evaluate these commutators, $[u_1, b]$, and, $[u_2, b]$, and replace them with their resultant formulas. To do this, we're going to replace unit squares, like, u_1^2, with equivalent forms that linearize them, e.g, $u_1^2 = -1 + 2u_{10}u_1$,

etc..., and use various conjugation and commutator rules, like, $[A,B]^* = [A,B^*] = [A^*,B] = -[A,B]$, which we've seen before, again here found useful. We start with modifying the formula expression for the b parameter.

$$
\begin{aligned}
b &= -(m^2u_1^2 - n^2u_2^2)(mu_1^* - nu_2^*)/|mu_1 - nu_2|^2 && (1.457)\\
&= -(m^2(-1+2u_{10}u_1) - n^2(-1+2u_{20}u_2))(mu_1^* - nu_2^*)/|d|^2 && (1.458)\\
&= -(\ n^2 - m^2 + 2m^2u_{10}u_1 - 2n^2u_{20}u_2\ )(mu_1^* - nu_2^*)/|d|^2 && (1.459)
\end{aligned}
$$

$$
\begin{aligned}
u_1\cdot b &= -u_1\cdot(\ n^2 - m^2 + 2m^2u_{10}u_1 - 2n^2u_{20}u_2\ )(mu_1^* - nu_2^*)/|d|^2 && (1.460)\\
&= -(\ (n^2 - m^2)\cdot u_1 + 2m^2u_{10}\cdot u_1\cdot u_1 - 2n^2u_{20}\cdot u_1\cdot u_2\ )(mu_1^* - nu_2^*)/|d|^2 && (1.461)\\
&= -(\ (n^2 - m^2 + 2m^2u_{10}\cdot u_1)\cdot u_1 - 2n^2u_{20}\cdot(u_2u_1 + [u_1,u_2])\ )(mu_1^* - nu_2^*)/|d|^2 && (1.462)
\end{aligned}
$$

$$
\begin{aligned}
&= -(\ n^2 - m^2 + 2m^2u_{10}\cdot u_1 - 2n^2u_{20}\cdot u_2\ )\cdot u_1\cdot(mu_1^* - nu_2^*)/|d|^2 && (1.463)\\
&\quad +2n^2u_{20}\cdot[u_1,u_2]\cdot(mu_1^* - nu_2^*)/|d|^2 && (1.464)
\end{aligned}
$$

$$
\begin{aligned}
u_2\cdot b &= -u_2\cdot(\ n^2 - m^2 + 2m^2u_{10}u_1 - 2n^2u_{20}u_2\ )(mu_1^* - nu_2^*)/|d|^2 && (1.465)\\
&= -(\ (n^2 - m^2)\cdot u_2 + 2m^2u_{10}\cdot u_2\cdot u_1 - 2n^2u_{20}\cdot u_2\cdot u_2\ )(mu_1^* - nu_2^*)/|d|^2 && (1.466)\\
&= -(\ (n^2 - m^2 - 2n^2u_{20}\cdot u_2)\cdot u_2 + 2m^2u_{10}\cdot(u_1u_2 + [u_2,u_1])\ )(mu_1^* - nu_2^*)/|d|^2 && (1.467)
\end{aligned}
$$

$$
\begin{aligned}
&= -(\ n^2 - m^2 + 2m^2u_{10}\cdot u_1 - 2n^2u_{20}\cdot u_2\ )\cdot u_2\cdot(mu_1^* - nu_2^*)/|d|^2 && (1.468)\\
&\quad -2m^2u_{10}\cdot[u_2,u_1]\cdot(mu_1^* - nu_2^*)/|d|^2 && (1.469)\\
& && (1.470)
\end{aligned}
$$

Now we need the commutators, $[\,u_1,d^*\,]$, and, $[\,u_2,d^*\,]$, and need to process the expression, $[\,u_1,u_2\,]\cdot(mu_1^* - mu_2^*)$, so that the commutator of the units, $[\,u_1,u_2\,]$, moves all the way to the right of the expression.

$$
\begin{aligned}
u_1\cdot d^* &= u_1\cdot(mu_1^* - nu_2^*) = (mu_1u_1^* - nu_1u_2^*) = (mu_1^*u_1 - n(u_2^*u_1 + [u_1,u_2^*])) && (1.471)\\
&= (mu_1^* - nu_2^*)u_1 - n[u_1,u_2^*] && (1.472)\\
&= (mu_1^* - nu_2^*)u_1 + n[u_1,u_2] && (1.473)\\
&= d^*u_1 + n[u_1,u_2] && (1.474)\\
&= d^*u_1 + [d,s]/(2m) && (1.475)\\
[\,u_1,d^*\,] &= [d,s]/(2m) && (1.476)
\end{aligned}
$$

$$
\begin{aligned}
u_2\cdot d^* &= u_2\cdot(mu_1^* - nu_2^*) = (mu_2u_1^* - nu_2u_2^*) = (m(u_1^*u_2 + [u_2,u_1^*]) - nu_2^*u_2) && (1.477)\\
&= (mu_1^* - nu_2^*)u_2 + m[u_2,u_1^*] && (1.478)\\
&= (mu_1^* - nu_2^*)u_2 + m[u_1,u_2] && (1.479)\\
&= d^*u_2 + m[u_1,u_2] && (1.480)\\
&= d^*u_1 + [d,s]/(2n) && (1.481)\\
[\,u_1,d^*\,] &= [d,s]/(2n) && (1.482)
\end{aligned}
$$

$$
\begin{aligned}
[u_1,u_2](mu_1^* - nu_2^*) &= (u_1u_2 - u_2u_1)(mu_1^* - nu_2^*) && (1.483)\\
&= (mu_1u_2u_1^* - nu_1u_2u_2^* - mu_2u_1u_1^* + nu_2u_1u_2^*) && (1.484)\\
&= (mu_1u_2u_1^* - nu_1 - mu_2 + nu_2u_1u_2^*)) && (1.485)\\
&= (mu_1(u_1^*u_2 + [u_2,u_1^*]) - nu_1 - mu_2 + nu_2(u_2^*u_1 + [u_1,u_2^*])) && (1.486)\\
&= (mu_1u_1^*u_2 + mu_1[u_2,u_1^*] - nu_1 - mu_2 + nu_2u_2^*u_1 + nu_2[u_1,u_2^*]) && (1.487)\\
&= (mu_2 - mu_1[u_2,u_1] - nu_1 - mu_2 + nu_1 - nu_2[u_1,u_2]) && (1.488)\\
&= (mu_2 + mu_1[u_1,u_2] - nu_1 - mu_2 + nu_1 - nu_2[u_1,u_2]) && (1.489)\\
&= (mu_1 - nu_2)[u_1,u_2] && (1.490)\\
\text{i.e.}\quad [u_1,u_2]\cdot d^* &= d\cdot[u_1,u_2]\ ,\quad \text{cf.}\rightarrow [d,s]\cdot d^* = d\cdot[d,s],\ \ [d,s] = 2[mu_1,nu_2] = 2mn[u_1,u_2]\ \ \text{etc..} && (1.491)
\end{aligned}
$$

Now we can finish the evaluation of the b commutators,

$$u_1 \cdot b = -(\, n^2 - m^2 + 2m^2 u_{10} \cdot u_1 - 2n^2 u_{20} \cdot u_2 \,) \cdot u_1 \cdot d^* / |d|^2 \tag{1.492}$$
$$+2n^2 u_{20} \cdot [u_1, u_2] \cdot (m u_1^* - n u_2^*)/|d|^2 \tag{1.493}$$

$$= -(\, n^2 - m^2 + 2m^2 u_{10} \cdot u_1 - 2n^2 u_{20} \cdot u_2 \,) \cdot d^* \cdot u_1 / |d|^2 \tag{1.494}$$
$$-(\, n^2 - m^2 + 2m^2 u_{10} \cdot u_1 - 2n^2 u_{20} \cdot u_2 \,) \cdot n[u_1, u_2]/|d|^2 \tag{1.495}$$
$$+2n^2 u_{20} \cdot (m u_1 - n u_2) \cdot [u_1, u_2]/|d|^2 \tag{1.496}$$

$$= b \cdot u_1 + [-n(\, n^2 - m^2 + 2m^2 u_{10} \cdot u_1 - 2n^2 u_{20} \cdot u_2 \,) + 2n^2 u_{20} \cdot (m u_1 - n u_2)] \cdot \frac{[u_1, u_2]}{|d|^2} \tag{1.497}$$

$$[u_1, b] = \frac{n}{|d|^2} \cdot [\, m^2 - n^2 - 2m(m u_{10} - n u_{20}) \cdot u_1 \,] \cdot [u_1, u_2] \tag{1.498}$$

$$u_2 \cdot b = -(\, n^2 - m^2 + 2m^2 u_{10} \cdot u_1 - 2n^2 u_{20} \cdot u_2 \,) \cdot u_2 \cdot d^* / |d|^2 \tag{1.499}$$
$$+2m^2 u_{10} \cdot [u_1, u_2] \cdot (m u_1^* - n u_2^*)/|d|^2 \tag{1.500}$$

$$= -(\, n^2 - m^2 + 2m^2 u_{10} \cdot u_1 - 2n^2 u_{20} \cdot u_2 \,) \cdot d^* \cdot u_2 / |d|^2 \tag{1.501}$$
$$-(\, n^2 - m^2 + 2m^2 u_{10} \cdot u_1 - 2n^2 u_{20} \cdot u_2 \,) \cdot m[u_1, u_2]/|d|^2 \tag{1.502}$$
$$+2m^2 u_{10} \cdot (m u_1 - n u_2) \cdot [u_1, u_2]/|d|^2 \tag{1.503}$$

$$= b \cdot u_2 + [-m(\, n^2 - m^2 + 2m^2 u_{10} \cdot u_1 - 2n^2 u_{20} \cdot u_2 \,) + 2m^2 u_{10} \cdot (m u_1 - n u_2)] \cdot \frac{[u_1, u_2]}{|d|^2} \tag{1.504}$$

$$[u_2, b] = \frac{m}{|d|^2} \cdot [\, m^2 - n^2 - 2n(m u_{10} - n u_{20}) \cdot u_2 \,] \cdot [u_1, u_2] \tag{1.505}$$

$$\tag{1.506}$$

now for the whole commutator bracket expression,

$$-|d|^2 \cdot \lambda^2 \cdot [\, -m(\lambda^2 - n^2) \cdot [\, u_1, b \,] + n(\lambda^2 - m^2) \cdot [\, u_2, b \,] \,] \tag{1.507}$$

$$= -|d|^2 \cdot \lambda^2 \cdot \begin{bmatrix} -m(\lambda^2 - n^2) \cdot \frac{n}{|d|^2} \cdot [\, m^2 - n^2 - 2m(m u_{10} - n u_{20}) \cdot u_1 \,] \cdot [u_1, u_2] \\ +n(\lambda^2 - m^2) \cdot \frac{m}{|d|^2} \cdot [\, m^2 - n^2 - 2n(m u_{10} - n u_{20}) \cdot u_2 \,] \cdot [u_1, u_2] \end{bmatrix} \tag{1.508}$$

$$= -\lambda^2 \cdot \begin{bmatrix} -m(\lambda^2 - n^2) \cdot n \cdot [\, m^2 - n^2 - 2m(m u_{10} - n u_{20}) \cdot u_1 \,] \\ +n(\lambda^2 - m^2) \cdot m \cdot [\, m^2 - n^2 - 2n(m u_{10} - n u_{20}) \cdot u_2 \,] \end{bmatrix} \cdot [u_1, u_2] \tag{1.509}$$

$$= mn\lambda^2 \cdot [\, (\lambda^2 - n^2)[\, m^2 - n^2 - 2m(m u_{10} - n u_{20}) u_1 \,] - (\lambda^2 - m^2)[\, m^2 - n^2 - 2n(m u_{10} - n u_{20}) u_2 \,] \,] \, [u_1, u_2]$$

$$= mn\lambda^2 \cdot [\, (m^2 - n^2)^2 - 2(m u_{10} - n u_{20})(\, (\lambda^2 - n^2) m u_1 - (\lambda^2 - m^2) n u_2 \,) \,] \, [u_1, u_2] \tag{1.510}$$

thus, so far, we have,

$$(c b^* (\lambda^2 - c) - \lambda^2 (\lambda^2 - c) b) \tag{1.511}$$

$$= \frac{d}{|d|^4} \cdot \begin{bmatrix} (mn u_2 d^* u_1 \cdot b^* d - |d|^2 \cdot \lambda^2 \cdot b) \cdot [\, 2\lambda^2 (m u_{10} - n u_{20}) - m(\lambda^2 - n^2) u_1 + n(\lambda^2 - m^2) u_2) \,] \\ +mn\lambda^2 \cdot [\, (m^2 - n^2)^2 - 2(m u_{10} - n u_{20})(\, m(\lambda^2 - n^2) u_1 - n(\lambda^2 - m^2) u_2 \,) \,] \cdot [u_1, u_2] \end{bmatrix} \tag{1.512}$$

We could also use the identities, $u_1[u_1, u_2] = [u_1, u_2] u_1^*$, and, $u_2[u_1, u_2] = [u_1, u_2] u_2^*$, to move the commutator, $[u_1, u_2]$, to the left of the inner bracket expression; e.g. $u_1[u_1, u_2] = -u_1[u_1, u_2]^* = -u_1(u_2^* u_1^* - u_1^* u_2^*) = -(u_1 u_2^* u_1^* - u_2^* u_1 u_1^*) = -[u_1, u_2^*] u_1^* = [u_1, u_2] u_1^*$, etc.., so this formula could also be written,

$$(cb^*(\lambda^2-c)-\lambda^2(\lambda^2-c)b)$$

$$= \frac{d}{|d|^4}\cdot\left[\begin{array}{c}(mnu_2d^*u_1\cdot b^*d-|d|^2\cdot\lambda^2\cdot b)\cdot[\ 2\lambda^2(mu_{10}-nu_{20})-m(\lambda^2-n^2)u_1+n(\lambda^2-m^2)u_2)\ ] \\ +mn\lambda^2\cdot[u_1,u_2]\cdot[\ (m^2-n^2)^2-2(mu_{10}-nu_{20})(\ m(\lambda^2-n^2)u_1^*-n(\lambda^2-m^2)u_2^*\ )\ ]\end{array}\right] \tag{1.513}$$

We still have to deal with that factor expression containing, b, b^*, so lets look at that. First, we review these;

$$s = mu_1+nu_2 \tag{1.514}$$
$$d = mu_1-nu_2 \tag{1.515}$$
$$mu_1 = (s+d)/2 \tag{1.516}$$
$$nu_2 = (s-d)/2 \tag{1.517}$$
$$sd = (mu_1+nu_2)(mu_1-nu_2) = m^2u_1^2-n^2u_2^2+nmu_2u_1-mnu_1u_2 \tag{1.518}$$
$$ds = (mu_1-nu_2)(mu_1+nu_2) = m^2u_1^2-n^2u_2^2-nmu_2u_1+mnu_1u_2 \tag{1.519}$$
$$\{d,s\} = ds+sd = 2(m^2u_1^2-n^2u_2^2) \tag{1.520}$$
$$[d,s] = ds-sd = 2mn(u_1u_2-u_2u_1) = 2mn[u_1,u_2] \tag{1.521}$$
$$sd = ds-[d,s] \tag{1.522}$$
$$ds = sd+[d,s] \tag{1.523}$$
$$b = -(m^2u_1^2-n^2u_2^2)(mu_1^*-nu_2^*)/|mu_1-nu_2|^2 = -(1/2)(sd+ds)d^*/|d|^2 \tag{1.524}$$
$$= -(1/2)\cdot(\ 2sd+[d,s]\ )d^*/|d|^2 \tag{1.525}$$
$$b^* = -(1/2)\cdot d(\ 2d^*s^*+[d,s]^*)/|d|^2 \tag{1.526}$$
$$= -(1/2)\cdot d(\ 2d^*s^*-[d,s]\ )/|d|^2 \tag{1.527}$$
$$b^*d = -(1/2)\cdot d(\ 2d^*s^*-[d,s]\ )d/|d|^2 \tag{1.528}$$
$$= -(1/2)\cdot d(\ 2d^*s^*d-[d,s]d\ )/|d|^2 \tag{1.529}$$
$$= -(1/2)\cdot d(\ 2d^*s^*d-d^*[d,s]\ )/|d|^2 \tag{1.530}$$
$$= -(1/2)\cdot dd^*(\ 2s^*d-[d,s]\ )/|d|^2 \tag{1.531}$$
$$= -(1/2)\cdot(\ 2s^*d-[d,s]\ ) \tag{1.532}$$
$$mnu_2d^*u_1 = mnu_2(mu_1^*-nu_2^*)u_1 = mn(mu_2u_1^*u_1-nu_2u_2^*u_1) = mn(mu_2-nu_1) \tag{1.533}$$

Let's fire up the next expression,

$$(mnu_2d^*u_1\cdot b^*d-|d|^2\cdot\lambda^2\cdot b) \tag{1.534}$$

$$= (mnu_2d^*u_1\cdot[\ -(1/2)(2s^*d-[d,s])\ ]-\lambda^2\cdot|d|^2\cdot[\ -(1/2)(2sd+[d,s])d^*/|d|^2\ ]) \tag{1.535}$$
$$= (mnu_2d^*u_1\cdot[\ -(1/2)(2s^*d-[d,s])\ ]-\lambda^2\cdot[\ -(1/2)(2sd+[d,s])d^*\ ]) \tag{1.536}$$
$$= (mnu_2d^*u_1\cdot[\ -(1/2)(2s^*d-[d,s])\ ]-\lambda^2\cdot[\ -(1/2)(2sdd^*+d[d,s])\ ]) \tag{1.537}$$
$$= (mnu_2d^*u_1\cdot[\ -s^*d+mn[u_1,u_2]\ ]+\lambda^2\cdot[\ sdd^*+dmn[u_1,u_2]\ ]) \tag{1.538}$$
$$= [\ -nu_2\cdot d^*\cdot mu_1\cdot s^*d+\lambda^2\cdot dd^*s\ ]+[\ mnu_2d^*u_1\cdot mn[u_1,u_2]+\lambda^2\cdot mnd[u_1,u_2]\ ] \tag{1.539}$$
$$= \left[\begin{array}{c}-(s-d)/2\cdot d^*\cdot(s+d)/2\cdot s^*d+\lambda^2\cdot dd^*s\ ] \\ +mn(mu_2-nu_1)\cdot mn[u_1,u_2]+\lambda^2\cdot mn(mu_1-nu_2)[u_1,u_2]\end{array}\right] \tag{1.540}$$

working on the (d,s) part of this last expression;

$$[-(s-d)/2\cdot d^*\cdot(s+d)/2\cdot s^*d+\lambda^2\cdot dd^*s] = (1/4)[-(sd^*-dd^*)(ss^*+ds^*)d+4\lambda^2\cdot dd^*s] \tag{1.541}$$
$$= (1/4)[-(sd^*|s|^2+sd^*ds^*-|d|^2|s|^2-|d|^2ds^*)d+4\lambda^2\cdot dd^*s] \tag{1.542}$$
$$= (1/4)[-(sd^*d|s|^2+sd^*ds^*d-|d|^2|s|^2d-|d|^2ds^*d)+4\lambda^2\cdot dd^*s] \tag{1.543}$$
$$= (1/4)\cdot|d|^2\cdot[-(s|s|^2+|s|^2d-|s|^2d-ds^*d)+4\lambda^2\cdot s] \tag{1.544}$$
$$= |d|^2\cdot[\ \lambda^2\cdot s-(s|s|^2-ds^*d)/4\ ] \tag{1.545}$$
$$\tag{1.546}$$

working out, (d^*s), and, $(s^*d) = (d^*s)^*$, we get,

$$\begin{aligned}(d^*s) &= (mu_1^* - nu_2^*)(mu_1 + nu_2) = (m^2 - n^2 + mnu_1^*u_2 - nmu_2^*u_1) &(1.547)\\ (d^*s)^* &= (mu_1^* + nu_2^*)(mu_1 - nu_2) = (m^2 - n^2 + mnu_2^*u_1 - nmu_1^*u_2) = (s^*d) &(1.548)\end{aligned}$$

continuing...

$$\begin{aligned}&(mnu_2d^*u_1 \cdot b^*d - |d|^2 \cdot \lambda^2 \cdot b)\\ &= \begin{bmatrix} |d|^2 \cdot [\ \lambda^2 \cdot s - (s|s|^2 - ds^*d)/4\ ] \\ +mn(mu_2 - nu_1) \cdot mn[u_1, u_2] + \lambda^2 \cdot mn(mu_1 - nu_2)[u_1, u_2] \end{bmatrix} &(1.549)\\ &= \begin{bmatrix} |d|^2 \cdot [\ \lambda^2 \cdot s - (s|s|^2 - ds^*d)/4\ ] \\ +mn[\ (\lambda^2 - n^2)mu_1 - (\lambda^2 - m^2)nu_2\ ] \cdot [u_1, u_2] \end{bmatrix} &(1.550)\\ &= \begin{bmatrix} |d|^2 \cdot [\ \lambda^2 \cdot s - (s|s|^2 - ds^*d)/4\ ] \\ +m^2n(\lambda^2 - n^2) \cdot u_1 \cdot [u_1, u_2] \\ -mn^2(\lambda^2 - m^2) \cdot u_2 \cdot [u_1, u_2] \end{bmatrix} &(1.551)\end{aligned}$$

We may now update our L.H.S factor,

$$\begin{aligned}&(cb^*(\lambda^2 - c) - \lambda^2(\lambda^2 - c)b)\\ &= \frac{d}{|d|^4} \cdot \begin{bmatrix} \begin{bmatrix} |d|^2 \cdot [\ \lambda^2 \cdot s - (s|s|^2 - ds^*d)/4\ ] \\ +mn \cdot m(\lambda^2 - n^2) \cdot [u_1, u_2]u_1^* \\ -mn \cdot n(\lambda^2 - m^2) \cdot [u_1, u_2]u_2^* \end{bmatrix} \cdot [\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2)\ ] \\ +mn\lambda^2 \cdot [u_1, u_2] \cdot [\ (m^2 - n^2)^2 - 2(mu_{10} - nu_{20})(\ m(\lambda^2 - n^2)u_1^* - n(\lambda^2 - m^2)u_2^*\ )\ ] \end{bmatrix} &(1.552)\end{aligned}$$

where we've used the identities, $u_1[u_1, u_2] = [u_1, u_2]u_1^*$, and, $u_2[u_1, u_2] = [u_1, u_2]u_2^*$, to prepare for combining expressions that match terms in the last line. Expanding the relevant fragment,

$$\begin{aligned}&\begin{bmatrix} +mn \cdot m(\lambda^2 - n^2) \cdot [u_1, u_2]u_1^* \\ -mn \cdot n(\lambda^2 - m^2) \cdot [u_1, u_2]u_2^* \end{bmatrix} \cdot [\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2)\ ] &(1.553)\\ &= +mn\lambda^2 \cdot [u_1, u_2] \cdot [\ +2(mu_{10} - nu_{20})(\ m(\lambda^2 - n^2)u_1^* - n(\lambda^2 - m^2)u_2^*\ )\ ] &(1.554)\\ &\quad +mn \cdot [u_1, u_2] \cdot [\ -m^2(\lambda^2 - n^2)^2 - n^2(\lambda^2 - m^2)^2 + mn(\lambda^2 - m^2)(\lambda^2 - n^2)(u_1^*u_2 + u_2^*u_1)\ ] &(1.555)\end{aligned}$$

The top line here, cancels part of the last line in the large bracketed expression, but a new term involving the commutator, $[u_1, u_2]$, appears. However, this new expression on the bottom line evaluates to a scalar times $[u_1, u_2]$, and putting, $\lambda = m$, and, $\lambda = n$, alternatively, reveals that it has just the right form to remove the remaining part of the last line in the large bracket. So, on the L.H.S of the constraint we have,

$$\begin{aligned}&(cb^*(\lambda^2 - c) - \lambda^2(\lambda^2 - c)b)\\ &= \frac{d}{|d|^4} \cdot \begin{bmatrix} [\ |d|^2 \cdot [\ \lambda^2 \cdot s - (s|s|^2 - ds^*d)/4\ ]\ ] \cdot [\ 2\lambda^2(mu_{10} - nu_{20}) - m(\lambda^2 - n^2)u_1 + n(\lambda^2 - m^2)u_2)\ ] \\ +mn\lambda^2 \cdot [u_1, u_2] \cdot [\ (m^2 - n^2)^2\ ] \\ +mn \cdot [u_1, u_2] \cdot [\ -m^2(\lambda^2 - n^2)^2 - n^2(\lambda^2 - m^2)^2 + mn(\lambda^2 - m^2)(\lambda^2 - n^2)(u_1^*u_2 + u_2^*u_1)\ ] \end{bmatrix} &(1.556)\end{aligned}$$

Now we have just enough information to tackle the proof. Recall again, our provisional unit direction function, $p(\lambda)$, which we artfully transformed into the simpler form for this analysis;

$$p(\lambda) \quad = \lambda \cdot (\lambda^2 - c^*) \cdot \left(\frac{cb^* \cdot (\lambda^2 - c) - \lambda^2(\lambda^2 - c) \cdot b}{(\lambda^4 - |c|^2) \cdot |\lambda^2 - c|^2} \right) \tag{1.557}$$

The requirement, $|p(\lambda)| = 1$, then means,

$$\begin{array}{lll} & \text{L.H.S} & \text{R.H.S} \\ |\lambda| \cdot |(cb^* \cdot (\lambda^2 - c) - \lambda^2(\lambda^2 - c) \cdot b)| & = & |\lambda^4 - |c|^2| \cdot |\lambda^2 - c| \end{array} \tag{1.558}$$

The method first tries to match, $\lambda = m$, so to see if this is always possible, we replace λ.

$$\text{L.H.S:} \quad \lambda = m \tag{1.559}$$
$$(cb^*(\lambda^2 - c) - \lambda^2(\lambda^2 - c)b)$$

$$= \frac{d}{|d|^4} \cdot \left[\begin{array}{r} \left[\ |d|^2 \cdot \left[\ m^2 \cdot s - (s|s|^2 - ds^*d)/4 \ \right] \ \right] \cdot \left[\ 2m^2(mu_{10} - nu_{20}) - m(m^2 - n^2)u_1 + n(m^2 - m^2)u_2) \ \right] \\ +mnm^2 \cdot [u_1, u_2] \cdot \left[\ (m^2 - n^2)^2 \ \right] \\ +mn \cdot [u_1, u_2] \cdot \left[\ -m^2(m^2 - n^2)^2 - n^2(m^2 - m^2)^2 + mn(m^2 - m^2)(m^2 - n^2)(u_1^* u_2 + u_2^* u_1) \ \right] \end{array} \right] \tag{1.560}$$

$$= \frac{d}{|d|^4} \cdot \left[\ \left[\ |d|^2 \cdot \left[\ m^2 \cdot s - (s|s|^2 - ds^*d)/2 \ \right] \ \right] \cdot \left[\ 2m^2(mu_{10} - nu_{20}) - m(m^2 - n^2)u_1 \ \right] \ \right]$$

$$= \frac{d}{|d|^2} \cdot \left[\ m^2 \cdot s - (s|s|^2 - ds^*d)/4 \ \right] \cdot \left[\ 2m^2(mu_{10} - nu_{20}) - m(m^2 - n^2)u_1 \ \right] \tag{1.561}$$

Working out the intermediate unresolved expression with (d, s), parameters,

$$m^2 \cdot s - (s|s|^2 - d(s^*d))/4 = m^2 \cdot s - (1/4) \left[\begin{array}{r} (mu_1 + nu_2)(m^2 + n^2 + mnu_1^* u_2 + mnu_2^* u_1) \\ -(mu_1 - nu_2)(m^2 - n^2 + mnu_2^* u_1 - mnu_1^* u_2) \end{array} \right] \tag{1.562}$$

$$= m^2 \cdot s - \frac{1}{4} \cdot \left[\begin{array}{r} +m^3u_1 + m^2nu_2 + mn^2u_1 + n^3u_2 + mn^2u_1 + m^2nu_2 + mn^2u_2u_1^*u_2 + m^2nu_1u_2^*u_1 \\ -m^3u_1 + m^2nu_2 + mn^2u_1 - n^3u_2 + mn^2u_1 + m^2nu_2 - mn^2u_2u_1^*u_2 - m^2nu_1u_2^*u_1 \end{array} \right] \tag{1.563}$$

$$= m^2 \cdot (mu_1 + nu_2) - \left[\ +mn^2u_1 + m^2nu_2 \ \right] \quad = \quad m(m^2 - n^2)u_1 \tag{1.564}$$

Hence,

$$\text{L.H.S:} \quad \lambda = m \tag{1.565}$$
$$|\lambda| \cdot |cb^*(\lambda^2 - c) - \lambda^2(\lambda^2 - c)b| = \tag{1.566}$$
$$|m| \cdot \frac{|d|}{|d|^2} \cdot |m(m^2 - n^2)| \cdot |u_1| \cdot |2m^2(mu_{10} - nu_{20}) - m(m^2 - n^2)u_1| \tag{1.567}$$
$$\text{R.H.S:} \quad \lambda = m \tag{1.568}$$
$$|\lambda^4 - |c|^2| \cdot |\lambda^2 - c| = \frac{|m^4 - m^2n^2|}{|mu_1 - nu_2|} \cdot | \ 2m^2(mu_{10} - nu_{20}) - m(m^2 - n^2)u_1 + n(m^2 - m^2)u_2 \ | \tag{1.569}$$

$$= \frac{|m^4 - m^2n^2|}{|mu_1 - nu_2|} \cdot | \ 2m^2(mu_{10} - nu_{20}) - m(m^2 - n^2)u_1 \ | \tag{1.570}$$

Match succeeds, L.H.S = R.H.S, since, $|u_1| = 1$, and, $|d|/|d|^2 = 1/|mu_1 - nu_2|$, $mu_1 \neq nu_2$, by problem specification, so the method is able to find the magnitude, $\lambda = m$, and the corresponding unit direction, $p(\lambda = m)$, given by,

$$p(\lambda = m) \quad = \frac{m \cdot (m^2 - c^*) \cdot d \cdot (m(m^2 - n^2)u_1)[2m^2(mu_{10} - nu_{20}) - m(m^2 - n^2)u_1]}{|d|^2(m^4 - m^2n^2)|d|^{-2}|2m^2(mu_{10} - nu_{20}) - m(m^2 - n^2)u_1|^2} \tag{1.571}$$
$$= \frac{(m^2 - c^*) \cdot d \cdot [2m^2(mu_{10} - nu_{20}) - m(m^2 - n^2)u_1] \cdot u_1}{|2m^2(mu_{10} - nu_{20}) - m(m^2 - n^2)u_1|^2} \tag{1.572}$$
$$= \frac{[2m^2(mu_{10} - nu_{20}) - m(m^2 - n^2)u_1]^* \cdot d^*d \cdot [2m^2(mu_{10} - nu_{20}) - m(m^2 - n^2)u_1] \cdot u_1}{|d|^2|2m^2(mu_{10} - nu_{20}) - m(m^2 - n^2)u_1|^2} \tag{1.573}$$
$$= \quad u_1 \tag{1.574}$$

Next the method tries to match, $\lambda = n$, so to see if this is always possible, we replace λ.

$$\text{L.H.S:}\quad \lambda = n \tag{1.575}$$
$$(cb^*(\lambda^2 - c) - \lambda^2(\lambda^2 - c)b)$$

$$= \frac{d}{|d|^4} \cdot \left[\begin{array}{r} \left[\ |d|^2 \cdot \left[\ n^2 \cdot s - (s|s|^2 - ds^*d)/4\ \right]\ \right] \cdot \left[\ 2n^2(mu_{10} - nu_{20}) - m(n^2 - n^2)u_1 + n(n^2 - m^2)u_2\ \right] \\ +mnn^2 \cdot [u_1, u_2] \cdot \left[\ (m^2 - n^2)^2\ \right] \\ +mn \cdot [u_1, u_2] \cdot \left[\ -m^2(n^2 - n^2)^2 - n^2(n^2 - m^2)^2 + mn(n^2 - m^2)(n^2 - n^2)(u_1^*u_2 + u_2^*u_1)\ \right] \end{array} \right] \tag{1.576}$$

$$= \frac{d}{|d|^4} \cdot \left[\ \left[\ |d|^2 \cdot \left[\ n^2 \cdot s - (s|s|^2 - ds^*d)/2\ \right]\ \right] \cdot \left[\ 2n^2(mu_{10} - nu_{20}) + n(n^2 - m^2)u_2\ \right]\ \right]$$

$$= \frac{d}{|d|^2} \cdot \left[\ n^2 \cdot s - (s|s|^2 - ds^*d)/4\ \right] \cdot \left[\ 2n^2(mu_{10} - nu_{20}) + n(n^2 - m^2)u_2\ \right] \tag{1.577}$$

Working out the intermediate unresolved expression with (d, s), parameters,

$$n^2 \cdot s - (s|s|^2 - d(s^*d))/4 = n^2 \cdot (mu_1 + nu_2) - \left[\ +mn^2u_1 + m^2nu_2\ \right] = n(n^2 - m^2)u_2 \tag{1.578}$$

Hence,

$$\text{L.H.S:}\quad \lambda = n \tag{1.579}$$
$$|\lambda| \cdot |cb^*(\lambda^2 - c) - \lambda^2(\lambda^2 - c)b| = \tag{1.580}$$
$$|n| \cdot \frac{|d|}{|d|^2} \cdot |n(n^2 - m^2)| \cdot |u_2| \cdot |2n^2(mu_{10} - nu_{20}) + n(n^2 - m^2)u_2| \tag{1.581}$$
$$\text{R.H.S:}\quad \lambda = n \tag{1.582}$$
$$|\lambda^4 - |c|^2| \cdot |\lambda^2 - c| = \frac{|n^4 - m^2n^2|}{|mu_1 - nu_2|} \cdot |\ 2n^2(mu_{10} - nu_{20}) - m(n^2 - n^2)u_1 + n(n^2 - m^2)u_2\ | \tag{1.583}$$
$$= \frac{|n^4 - m^2n^2|}{|mu_1 - nu_2|} \cdot |\ 2n^2(mu_{10} - nu_{20}) + n(n^2 - m^2)u_2\ | \tag{1.584}$$

Match succeeds, L.H.S = R.H.S, since, $|u_1| = 1$, and, $|d|/|d|^2 = 1/|mu_1 - nu_2|$, $mu_1 \neq nu_2$, by problem specification, so the method is able to find the magnitude, $\lambda = n$, and the corresponding unit direction, $p(\lambda = n)$, given by,

$$p(\lambda = n) = \frac{n \cdot (n^2 - c^*) \cdot d \cdot (n(n^2 - m^2)u_2)[2n^2(mu_{10} - nu_{20}) + n(n^2 - m^2)u_2]}{|d|^2(n^4 - m^2n^2)|d|^{-2}|2n^2(mu_{10} - nu_{20}) + n(n^2 - m^2)u_2|^2} \tag{1.585}$$
$$= \frac{(n^2 - c^*) \cdot d \cdot [2n^2(mu_{10} - nu_{20}) + n(n^2 - m^2)u_2] \cdot u_2}{|2n^2(mu_{10} - nu_{20}) + n(n^2 - m^2)u_2|^2} \tag{1.586}$$
$$= \frac{[2n^2(mu_{10} - nu_{20}) + n(n^2 - m^2)u_2]^* \cdot d^*d \cdot [2n^2(mu_{10} - nu_{20}) + n(n^2 - m^2)u_2] \cdot u_2}{|d|^2|2n^2(mu_{10} - nu_{20}) + n(n^2 - m^2)u_2|^2} \tag{1.587}$$
$$= u_2 \tag{1.588}$$

Q.E.D

This shows that, *in principle*, the *two-hand two-step* method, used for finding solutions to the quadradic equation of the form, $aq^2 + bq + c = 0$, should have no problems in the case of "*Different magnitudes, different directions*," and by numerically searching for the roots with the constraint, $|p(\lambda)| = 1$, should pick out both roots with equal ease.

`Corrollary:` Since, $a^{-1}c = mn(mu_1 - nu_2)u_2(mu_1^* - nu_2^*)u_1/|mu_1 - nu_2|^2$, and hence, $|a^{-1}c| = mn$, we have an expression for the product of the magnitudes of the two roots. But, this is the same expression that evaluates the square, λ^2, of the critical point, between the two distinct roots, $\lambda = \sqrt{|c|/|a|}$. Thus, the critical point is actually an *average* of the moduli; it is the *geometric mean* of the two root magnitudes. Since, this point is always known, and determined by the parameters, (a, c), we only really need to find one root magnitude to the quadratic equation, using numerical methods, the second root is completely determined by the first root and the critical point.

$$\lambda_0 = \sqrt{\frac{|c|}{|a|}}, \qquad \lambda_1\lambda_2 = |a^{-1}c| = \frac{|c|}{|a|} = \lambda_0^2, \qquad \lambda_0 = \sqrt{\lambda_1\lambda_2}, \qquad \lambda_2 = \frac{\lambda_0^2}{\lambda_1} = \frac{|a^*c|}{|a|^2\lambda_1} \tag{1.589}$$

If we order the roots, by convention, so that, $\lambda_1 \leq \lambda_2$, then the larger root is determined by the smaller root, and the interval, $\lambda_2 - \lambda_1$, is a function of just one root magnitude.

$$\texttt{width} \;\; = \;\; \lambda_2 - \lambda_1 = \frac{|a^* c|}{|a|^2} \cdot \frac{1}{\lambda_1} - \lambda_1 \tag{1.590}$$

This is the "`width`" of the curve, $f(\lambda) = p(\lambda)p(\lambda)^* - 1$, at the axis, whether the curve is *natural spike* or *hill top*. If we could find another expression for this width, using the formal parameters, $w = w(a, b, c)$, then we'd be able to set up a simpler "real valued" quadratic equation in the first root magnitude, λ_1, and so completely solve the problem with expressions in the usual form of *radicals*.

$$w(a, b, c) = \frac{|a^* c|}{|a|^2} \cdot \frac{1}{\lambda_1} - \lambda_1, \qquad \rightarrow \qquad \therefore \quad \lambda_1^2 + \lambda_1 \cdot w(a, b, c) - \frac{|a^* c|}{|a|^2} = 0 \tag{1.591}$$

One useful result of this fact, i.e. $\lambda_2 = \lambda_0^2/\lambda_1$, is that we don't really need to determine what upper bound to use in setting the search interval for the numerical procedure. The first root, λ_1, is easily determined by setting the search interval to be, $[\, 0, \lambda_0 - 10^{-5} \,]$, i.e., by subtracting a small number from it's upper bound, λ_0. However, what limit should we use for the larger root, $[\, \lambda_0 + 10^{-5}, \, ? \,]$. Theoretically, it's infinity, $[\lambda_0 + 10^{+5}, +\infty]$! We've set this upper bound to, 10^{23}, in our code, which, for all practical purposes, is probably just fine. Still, that's a long way from infinity. But, this limit depends on the numerical software and the computing machine floating point capabilities, so could be set higher in some computing environments. By searching only for the first root, we avoid this upper limit problem in real production coding. However, for educational purposes, finding the two roots, *then* confirming that the average of their product is the critical point, provides an additional verification that the calculation method works. It allows us to prove these relations, rather than assume them. Of course, there is the *inverse* problem, of how close to 0 the computing machine can actually get, which is related to the reciprocal of the largest number again, but at least we can set the lowest bound to, 0, in our code, while we can't usually set the highest bound at, $+\infty$.

Well, we've seen, in our graphical plots, that the linear parameter, b, determines the "width" of the curve at the axis. This suggests we consider that other formula we have, $a^{-1}b = -(m^2u_1^2 - n^2u_2^2)(mu_1^* - nu_2^*)/|mu_1 - nu_2|^2$, to see if we can establish a second relation that might help to determine the first root. Let's define the magnitude of this to be, Λ, so that,

$$\Lambda = |a^{-1}b| = \left| \frac{-(m^2u_1^2 - n^2u_2^2)(mu_1^* - nu_2^*)}{|mu_1 - nu_2|^2} \right| = \frac{|m^2u_1^2 - n^2u_2^2|}{|mu_1 - nu_2|} = \sqrt{\frac{(m^2u_1^2 - n^2u_2^2)(m^2u_1^{*2} - n^2u_2^{*2})}{(mu_1 - nu_2)(mu_1^* - nu_2^*)}}$$

$$\therefore \qquad \Lambda^2 \cdot [\, m^2 + n^2 - mn(u_1u_2^* + u_2u_1^*) \,] = [\, m^4 + n^4 - m^2n^2(u_1^2u_2^{*2} + u_2^2u_1^{*2}) \,] \tag{1.592}$$

Then, setting, $\lambda^2 = mn$, so that, $n = \lambda^2/m$, rearranging, we have a 4th order polynomial in the square, m^2, of the magnitude of the smaller root,

$$(m^2)^4 - (m^2)^3 \cdot \Lambda^2 + (m^2)^2 \cdot \lambda^2[\, \Lambda^2(u_1u_2^* + u_2u_1^*) - \lambda^2(u_1^2u_2^{*2} + u_2^2u_1^{*2}) \,] - (m^2) \cdot \Lambda^2\lambda^4 + \lambda^8 = 0 \tag{1.593}$$

Now, if only we could determine the two scalar quantities, $(u_1u_2^* + u_2u_1^*)$, and, $(u_1^2u_2^{*2} + u_2^2u_1^{*2})$, as functions of the formal parameters, (a, b, c), we'd have the solution immediately, given that 4th order polynomials in real variables have a well known closed form solution expressed in radicals. If we let, $\lambda_1 = m$, and consider the width equation;

$$m^2 + m \cdot w(a, b, c) - \lambda^2 = 0, \;\; \rightarrow \;\; m \cdot w(a, b, c) = \lambda^2 - m^2, \;\; \rightarrow \;\; m^2 \cdot w(a, b, c)^2 = (\lambda^2 - m^2)^2 = \lambda^4 - 2\lambda^2m^2 + m^4$$

rearranging and squaring,

$$(m^2)^2 - (m^2) \cdot [\, 2\lambda^2 + w(a, b, c)^2 \,] + \lambda^4 = 0 \tag{1.594}$$

$$[\, (m^2)^2 - (m^2) \cdot [\, 2\lambda^2 + w(a, b, c)^2 \,] + \lambda^4 \,]^2 = 0 \tag{1.595}$$

$$(m^2)^4 - (m^2)^3 \cdot 2[\, 2\lambda^2 + w(a, b, c)^2 \,] + (m^2)^2 \cdot [\, 2\lambda^4 + [\, 2\lambda^2 + w(a, b, c)^2 \,]^2 \,] - (m^2) \cdot 2\lambda^4[\, 2\lambda^2 + w(a, b, c)^2 \,] + \lambda^8 = 0$$

we find a very similar 4th order polynomial in the square, m^2, magnitude again. The first root magnitude, $\lambda_1 = m$, must be a solution to both of these 4th order equations simultaneously.

The first obvious suggestive idea, is to just match up the coefficients of the two 4th-order polynomials, and see if we get a valid formula for the width, $w(a,b,c)$. So we try this,

$$\text{coef. of } (m^2)^3\text{:} \qquad \Lambda^2 = 2[\,2\lambda^2 + w(a,b,c)^2\,] \tag{1.596}$$

$$\text{coef. of } (m^2)^2\text{:} \qquad \lambda^2[\,\Lambda^2(u_1u_2^* + u_2u_1^*) - \lambda^2(u_1^2u_2^{*2} + u_2^2u_1^{*2})\,] = [\,2\lambda^4 + [\,2\lambda^2 + w(a,b,c)^2\,]^2\,] \tag{1.597}$$

$$\text{coef. of } (m^2)^1\text{:} \qquad \Lambda^2\lambda^4 = 2\lambda^4[\,2\lambda^2 + w(a,b,c)^2\,] \tag{1.598}$$

$$[\,4\lambda^2 + 2w(a,b,c)^2\,] = \Lambda^2 \tag{1.599}$$

$$\sqrt{2}\cdot w(a,b,c) = \sqrt{\Lambda^2 - 4\lambda^2} \;=\; \sqrt{|a^{-1}b|^2 - 4|a^{-1}c|} \;=\; \sqrt{\frac{|b|^2 - 4|ac|}{|a|^2}} = \frac{\sqrt{|b|^2 - 4|ac|}}{|a|} \tag{1.600}$$

$$\lambda^2[\,\Lambda^2(u_1u_2^* + u_2u_1^*) - \lambda^2(u_1^2u_2^{*2} + u_2^2u_1^{*2})\,] = [\,2\lambda^4 + \frac{\Lambda^4}{4}\,] \tag{1.601}$$

$$\texttt{cf.:} \qquad ax^2 + bx + c = 0, \;\rightarrow\; x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}, \;\; w = \lambda_2 - \lambda_1 = \sqrt{\frac{b^2 - 4ac}{a^2}}, \qquad x, a, b, c, w \in \mathbb{R} \tag{1.602}$$

Although, at first glance, it appears things are working out, and we obtain a somewhat familiar quadratic form for the width corresponding to real number quadratics, on closer inspection two problems emerge. First, we have an extra factor of $\sqrt{2}$, that messes up the actual measure of the "`width`," when we restrict the quaternion formula to the real number domain. Second, the particular discriminant that appears under the squareroot, $(|b|^2 - 4|ac|)$, is often negative, producing an imaginary measure for the width, which is a real value measure of the difference in magnitudes, $w = \lambda_2 - \lambda_1$. However, the idea that the two polynomials are linked in some way, can't be far off. The next obvious idea, is that, perhaps, the quadratic expression for the width, might just be "one factor" in the original 4th-order polynomial; thus, enabling, $\lambda_1 = m$, to still be a solution to both equations simultaneously. So, rather than squaring this quadratic expression to get the corresponding 4th-degree expression, we assume we're missing some detail that is probably encapsulated in another quadratic expression with some other, as yet, unknown function, $g(a,b,c)$. So, we try constructing the 4th-degree the following way instead.

$$[\,(m^2)^2 - (m^2)\cdot[\,2\lambda^2 + w(a,b,c)^2\,] + \lambda^4\,]\cdot[\,(m^2)^2 - (m^2)\cdot g(a,b,c) + \lambda^4\,] = 0 \tag{1.603}$$

$$(m^2)^4 - (m^2)^3\cdot(\,g + [\,2\lambda^2 + w^2\,]\,) + (m^2)^2\cdot[\,2\lambda^4 + [\,2\lambda^2 + w^2\,]\cdot g\,] - (m^2)\cdot\lambda^4(\,g + [\,2\lambda^2 + w^2\,]\,) + \lambda^8 = 0 \tag{1.604}$$

$$\Lambda^2 = (\,g + [\,2\lambda^2 + w^2\,]\,) \tag{1.605}$$

$$\lambda^2[\,\Lambda^2(u_1u_2^* + u_2u_1^*) - \lambda^2(u_1^2u_2^{*2} + u_2^2u_1^{*2})\,] = [\,2\lambda^4 + [\,2\lambda^2 + w^2\,]\cdot g\,] \tag{1.606}$$

$$\Lambda^2\lambda^4 = \lambda^4(\,g + [\,2\lambda^2 + w^2\,]\,) \tag{1.607}$$

Now, on comparing terms, we can find the relationship between that unknown function, $g(a,b,c)$, and the width, $w(a,b,c)$. Then, using this to eliminate the unknown detail, we arrive at an alternative expression for the width, w;

$$g(a,b,c) = \Lambda^2 - [\,2\lambda^2 + w(a,b,c)^2\,] \tag{1.608}$$

$$\lambda^2[\,\Lambda^2(u_1u_2^* + u_2u_1^*) - \lambda^2(u_1^2u_2^{*2} + u_2^2u_1^{*2})\,] = [\,2\lambda^4 + [\,2\lambda^2 + w^2\,]\cdot(\Lambda^2 - [\,2\lambda^2 + w^2\,])\,] \tag{1.609}$$

$$\therefore \qquad (w^2)^2 - (w^2)\cdot[\Lambda^2 - 4\lambda^2] - \lambda^2\Lambda^2[2 - (u_1u_2^* + u_2u_1^*)] + \lambda^4[2 - (u_1^2u_2^{*2} + u_2^2u_1^{*2})] = 0 \tag{1.610}$$

So, now we have obtained a quadratic equation in the square width, w^2, and all of the coefficients here are scalars, just as we expect the solution, w^2, and, w, to be scalar real value numbers. Hence the solution is the usual *completing the squares formula* result,

$$w^2 = \frac{[\Lambda^2 - 4\lambda^2] \pm \sqrt{(\Lambda^2 - 4\lambda^2)^2 - 4(-\lambda^2\Lambda^2[2 - (u_1u_2^* + u_2u_1^*)] + \lambda^4[2 - (u_1^2u_2^{*2} + u_2^2u_1^{*2})])}}{2} \tag{1.611}$$

Numerical testing reveals this to be the correct formula for the width, $w(a,b,c)$, once we know the parameters, $\Lambda, \lambda, u_1, u_2$, so let's put back in the formal parameters, (a,b,c), where we can; i.e., $\Lambda = |b|/|a|$, and, $\lambda^2 = |c|/|a|$, we

get;

$$w^2 = \frac{[|b|^2 - 4|a^*c|] \pm \sqrt{(|b|^2 - 4|a^*c|)^2 - 4(-|a^*c||b|^2[2 - (u_1u_2^* + u_2u_1^*)] + |a^*c|^2[2 - (u_1^2u_2^{*2} + u_2^2u_1^{*2})])}}{2|a|^2} \tag{1.612}$$

Now, when the parameters are real valued, $a, b, c \in \mathbb{R}$, and the roots are real valued, so, $u_1 = \pm 1, u_2 = \pm 1$, the scalars, $(u_1u_2^* + u_2u_1^*) = \pm 2$, and, $(u_1^2u_2^{*2} + u_2^2u_1^{*2}) = +2$, and the formula reduces to,

$$(u_1, u_2) = (+1, +1),\ (-1, -1), \qquad\qquad (u_1, u_2) = (+1, -1),\ (-1, +1) \tag{1.613}$$

$$w^2 = \frac{[|b|^2 - 4|ac|] \pm \sqrt{(|b|^2 - 4|ac|)^2}}{2|a|^2}\ ,\ \text{ or, }\ w^2 = \frac{[|b|^2 - 4|ac|] \pm \sqrt{(|b|^2 + 4|ac|)^2}}{2|a|^2} \tag{1.614}$$

$$= \frac{[|b|^2 - 4|ac|] \pm (|b|^2 - 4|ac|)}{2|a|^2}\ , \qquad \text{or,} \qquad = \frac{[|b|^2 - 4|ac|] \pm (|b|^2 + 4|ac|)}{2|a|^2} \tag{1.615}$$

$$= \frac{2[|b|^2 - 4|ac|]}{2|a|^2} = \frac{[|b|^2 - 4|ac|]}{|a|^2}\ ,\ \text{ or, } = \frac{2|b|^2}{2|a|^2} = \frac{|b|^2}{|a|^2} \tag{1.616}$$

$$w(a, b, c) = \sqrt{\frac{|b|^2 - 4|ac|}{|a|^2}} = \frac{\sqrt{|b|^2 - 4|ac|}}{|a|}\ \checkmark\ ,\ \text{ or, } w(a, b, c) = \frac{|b|}{|a|}\ \ ? \tag{1.617}$$

And we obtain the correct formula, without the mysterious $\sqrt{2}$ factor, that messed things up before, on the previous attempt. Well, at least it's the right formula when, $ac > 0$. But, what happened to the width, when, $ac < 0$? We expect to see the discriminant become, $(|b|^2 + 4|ac|)$, but instead we arrive at the weird result, $w = |b|/|a|$. In real algebra, when the roots have the same sign, $(u_1, u_2) = (+1, +1), (-1, -1)$, we know this can only happen when, $ac > 0$, since then, $|\sqrt{b^2 - 4ac}| < |b|$, and the result of the squareroot can never overcome the value of the $-b$ parameter, to change the sign. So, the signs have to be the same. That part worked out well, in this quaternion formula. But, when the signs differ, $(u_1, u_2) = (+1, -1), (-1, +1)$, we must have, $ac < 0$, and thus need to see the squareroot part become effectively, $\sqrt{|b|^2 + 4|ac|}$. Something is obviously missing in our analysis here. However, we recall that unknown function, $g(a, b, c)$, that we had to include to arrive at this point, and which we just eliminated so that we could find this version of the width, $w(a, b, c)$. Perhaps, that g-factor has the other part of the width we need to complete this work. Since we just picked that function arbitrarily, let's investigate what we need to get things to work out right. We need another width function, $W = W(a, b, c)$, similar to the one we found, $w = w(a, b, c)$, but with complementary properties. Looking at the formulas above, it's clear that what we need to see is this,

$$W^2 = \frac{[|b|^2 + 4|ac|] \pm \sqrt{(|b|^2 + 4|ac|)^2}}{2|a|^2} \tag{1.618}$$

$$W^2 - w^2 = \frac{|b|^2 + 4|ac|}{2|a|^2} - \frac{|b|^2 - 4|ac|}{2|a|^2} = \frac{8|ac|}{2|a|^2} = 4\lambda^2 \ \rightarrow\ W^2 = 4\lambda^2 + w^2 \tag{1.619}$$

$$\text{but,} \qquad g = \Lambda^2 - [\,2\lambda^2 + w^2\,] \tag{1.620}$$

$$\therefore \qquad g = \Lambda^2 + 2\lambda^2 - [\,4\lambda^2 + w^2\,] = \Lambda^2 + [\,2\lambda^2 - W^2\,] \tag{1.621}$$

This effectively means that our original 4th-degree polynomial factors into the following,

$$\big[\,(m^2)^2 - (m^2)\cdot[\,2\lambda^2 + w(a, b, c)^2\,] + \lambda^4\,\big]\cdot\big[\,(m^2)^2 - (m^2)\cdot[\,\Lambda^2 + 2\lambda^2 - W(a, b, c)^2\,] + \lambda^4\,\big] = 0 \tag{1.622}$$

where the two functions, $w = w(a, b, c)$, and, $W = W(a, b, c)$, are both measures of the width, i.e. the separation, $\lambda_2 - \lambda_1$, or width of the function, $f(\lambda) = p(\lambda)p(\lambda)^* - 1$, at the axis. They complement each other, the one providing the width for one set of cases, while the other the width for the complementary remaining set. Now we can write,

$$W^2 = \frac{[|b|^2 + 4|a^*c|] \pm \sqrt{(|b|^2 - 4|a^*c|)^2 - 4(-|a^*c||b|^2[2 - (u_1u_2^* + u_2u_1^*)] + |a^*c|^2[2 - (u_1^2u_2^{*2} + u_2^2u_1^{*2})])}}{2|a|^2} \tag{1.623}$$

And this function will find the width when the other doesn't; at least, in the real valued cases. But, we still have to determine the formal parameter, (a, b, c), expressions for the two scalars, $(u_1u_2^* + u_2u_1^*)$, and, $(u_1^2u_2^{*2} + u_2^2u_1^{*2})$.

Otherwise, we have to solve the quadratic equation problem first, before we can use this width formula, which sort of defeats the whole purpose. This remains an open question.

Opps! Something happened! Our train of thought went awire. There's actually nothing wrong with our width function, $w = w(a, b, c)$. We really didn't need to go off to find that other width function, $W = W(a, b, c)$. However, we wanted to record here, as closely as possible, the sequence of thinking events that led us to formulate these two functions, $w = w(a, b, c)$, and, $W = W(a, b, c)$. The last few paragraphs record faithfully what happened in our brain cells, while working on this problem. Let's then see, what caused the confusion in our mind, that led us to think we had an error, on the one hand, and have the need to go find another width function, on the other hand, and what is it exactly that we discovered here that may be useful.

Consider the corresponding quadratic equation problem in the real domain. Let's examine this more carefully.

$$ax^2 + bx + c = 0 \ , \quad x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \ , \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \tag{1.624}$$

Let's consider the two complementary cases, $ac > 0$, and, $ac < 0$.

$$ac > 0 \ , \quad x_1 = \frac{-b + \sqrt{|b|^2 - 4|ac|}}{2 \cdot \mathtt{sign}[a] \cdot |a|} \ , \quad x_2 = \frac{-b - \sqrt{|b|^2 - 4|ac|}}{2 \cdot \mathtt{sign}[a] \cdot |a|} \ , \quad x_1 - x_2 = \frac{+\sqrt{|b|^2 - 4|ac|}}{\mathtt{sign}[a] \cdot |a|} \tag{1.625}$$

$$ac < 0 \ , \quad x_1 = \frac{-b + \sqrt{|b|^2 + 4|ac|}}{2 \cdot \mathtt{sign}[a] \cdot |a|} \ , \quad x_2 = \frac{-b - \sqrt{|b|^2 + 4|ac|}}{2 \cdot \mathtt{sign}[a] \cdot |a|} \ , \quad x_1 - x_2 = \frac{+\sqrt{|b|^2 + 4|ac|}}{\mathtt{sign}[a] \cdot |a|} \tag{1.626}$$

$$\tag{1.627}$$

We have the measure of the "interval" in real numbers, $x_1 - x_2$, which has a width.

$$|x_1 - x_2| = \frac{+\sqrt{|b|^2 - 4|ac|}}{|a|} \quad \text{=OR=} \quad \frac{+\sqrt{|b|^2 + 4|ac|}}{|a|} \tag{1.628}$$

But, our width, $w = \lambda_2 - \lambda_1$, is not, $w = |x_1 - x_2|$, it should be, $w = ||x_1| - |x_2||$! So, let's compute this.

$$ac > 0, \ b^2 > 4ac: \qquad w = ||x_1| - |x_2|| = \sqrt{x_1^2 + x_2^2 - 2|x_1||x_2|} = \sqrt{x_1^2 + x_2^2 - 2|x_1 x_2|} \tag{1.629}$$

$$= \sqrt{\frac{b^2 - 2b\sqrt{|b|^2 - 4|ac|} + (|b|^2 - 4|ac|)}{4|a|^2} + \frac{b^2 + 2b\sqrt{|b|^2 - 4|ac|} + (|b|^2 - 4|ac|)}{4|a|^2} - 2 \cdot \frac{b^2 - (|b|^2 - 4|ac|)}{4|a|^2}} = \frac{+\sqrt{|b|^2 - 4|ac|}}{|a|} \tag{1.630}$$

$$ac < 0, \qquad : \qquad w = ||x_1| - |x_2|| = \sqrt{x_1^2 + x_2^2 - 2|x_1||x_2|} = \sqrt{x_1^2 + x_2^2 - 2|x_1 x_2|} \tag{1.631}$$

$$= \sqrt{\frac{b^2 - 2b\sqrt{|b|^2 + 4|ac|} + (|b|^2 + 4|ac|)}{4|a|^2} + \frac{b^2 + 2b\sqrt{|b|^2 + 4|ac|} + (|b|^2 + 4|ac|)}{4|a|^2} - 2 \cdot \frac{(|b|^2 + 4|ac|) - b^2}{4|a|^2}} = \frac{|b|}{|a|} \quad \checkmark \tag{1.632}$$

So, our width, $w = w(a, b, c)$, was actually right, in all cases, all the time! The confusion between the width of the interval in real numbers, and the width of the interval in magnitudes, sent us off exploring other alternatives. But, we need not have worried, we had the complete solution in hand all along. However, our new width function, $W = W(a, b, c)$, does just what the first width, $w = w(a, b, c)$, didn't. It helps to compute the width of the interval in real numbers, and actually computes the sum of the magnitudes in quaternions. So, what we have found, is two functions, that compute the "*difference*" and the "*sum*" of the magnitudes,

$$w(a, b, c) = \lambda_2 - \lambda_1 \ , \qquad W(a, b, c) = \lambda_1 + \lambda_2 \tag{1.633}$$

and when the problem domain is restricted to real numbers, these then measure the width of the interval between the magnitudes, and the width of the interval of real valued solutions. And consequently, of course,

$$\lambda_1 \ = \ (W - w)/2 \ , \qquad \lambda_2 \ = \ (W + w)/2 \tag{1.634}$$

So, the magnitudes, λ_1, λ_2, can be determined by an expression in *radicals*, if only those two scalars, $u_1 u_2^* + u_2 u_1^*$, and, $u_1^2 u_2^{*2} + u_2^2 u_1^{*2}$, can be determined, as functions of, (a, b, c). Then, the directions, $p_1 = p(\lambda_1), \ p_2 = p(\lambda_2)$, can be determined from the magnitudes, and the complete solutions, $q_1 = \lambda_1 p_1, \ q_2 = \lambda_2 p_2$, can be determined, all in the usual explicit closed form radical formula expressions. Just need to find those two scalar expressions! This might be an impossible task, however. But, a proof either way is currently wanting.

`Case 4 - Vanishing Denominator:`
$|\lambda^4|a|^2 - |c|^2| = 0.$

The denominator vanishes when, $\lambda = \sqrt{|c|/|a|}$, $|a| \neq 0$, and from the previous cases, we notice that this is where $(\lambda^4 - m^2n^2) = 0$, where, $\lambda = m$, and, $\lambda = n$, are the two magnitudes. Hence, vanishing only occurs at the actual root, when the magnitudes are the same, $m = n$. Otherwise, the zero is generally between the two roots, and doesn't affect our ability to find the solutions. So, only in the case of *same magnitudes* do we usually need to be concerned with denominator zero points. It can happen in quaternions, however, that the distinct magnitudes are so close together, that there is ambiguity between the case of *same magnitudes* and *different magnitudes*, resulting from the numerical imprecision on input parameters, as seen in a previous example above. In this case, the user must decide whether his "*intent*" trumps the "*residual*" evidence, in considering his best choice for solution method.

Consider the provisional unit direction function, $p(\lambda)$,

$$p(\lambda) \quad = \quad \frac{\lambda^5(\lambda^4 a^* a a^* - c^* a c^*)(c b^* - \lambda^2 b a^*)a + \lambda(c^* c c^* - \lambda^4 a^* c a^*)(c b^* - \lambda^2 b a^*)c}{(\lambda^4|a|^2 - |c|^2) \cdot |\lambda^2 a - c|^2 \cdot |\lambda^2 a + c|^2} \tag{1.635}$$

#1st: we differentiate once..

$$P(\lambda) \quad = \quad \begin{pmatrix} 5\lambda^4(\lambda^4 a^* a a^* - c^* a c^*)(c b^* - \lambda^2 b a^*)a + \lambda^5(4\lambda^3 a^* a a^*)(c b^* - \lambda^2 b a^*)a \\ +\lambda^5(\lambda^4 a^* a a^* - c^* a c^*)(-2\lambda b a^*)a + (c^* c c^* - \lambda^4 a^* c a^*)(c b^* - \lambda^2 b a^*)c \\ +\lambda(-4\lambda^3 a^* c a^*)(c b^* - \lambda^2 b a^*)c + \lambda(c^* c c^* - \lambda^4 a^* c a^*)(-2\lambda b a^*)c \\ \hline (4\lambda^3|a|^2)(\lambda^4|a|^2 - \lambda^2(ca^* + ac^*) + |c|^2)(\lambda^4|a|^2 + \lambda^2(ca^* + ac^*) + |c|^2) \\ +(\lambda^4|a|^2 - |c|^2)(4\lambda^3|a|^2 - 2\lambda(ca^* + ac^*))(\lambda^4|a|^2 + \lambda^2(ca^* + ac^*) + |c|^2) \\ +(\lambda^4|a|^2 - |c|^2)(\lambda^4|a|^2 - \lambda^2(ca^* + ac^*) + |c|^2)(4\lambda^3|a|^2 + 2\lambda(ca^* + ac^*)) \end{pmatrix} \tag{1.636}$$

```
function P=psolqvh(r)     % ver:  1.0(?),  (aqq + bq + c = 0)
global a b c;
np = 5*r^4*(r^4*a'*a*a' - c'*a*c')*(c*b' - r^2*b*a')*a;
np = np + r^5*(4*r^3*a'*a*a')*(c*b' - r^2*b*a')*a;
np = np + r^5*(r^4*a'*a*a' - c'*a*c')*(-2*r*b*a')*a;
np = np + (c'*c*c' - r^4*a'*c*a')*(c*b' - r^2*b*a')*c;
np = np + r*(-4*r^3*a'*c*a')*(c*b' - r^2*b*a')*c;
np = np + r*(c'*c*c' - r^4*a'*c*a')*(- 2*r*b*a')*c;
dp = trace((4*r^3*a*a')*(r^4*a*a' - r^2*(c*a' + a*c') + c*c')*(r^4*a*a' + r^2*(c*a' + a*c') + c*c'))/2;
dp = dp + trace((r^4*a*a' - c*c')*(4*r^3*a*a' - 2*r*(c*a' + a*c'))*(r^4*a*a' + r^2*(c*a' + a*c') + c*c'))/2;
dp = dp + trace((r^4*a*a' - c*c')*(r^4*a*a' - r^2*(c*a' + a*c') + c*c')*(4*r^3*a*a' + 2*r*(c*a' + a*c')))/2;
p = np/dp;
P = [ real(p(1,1)) , imag(p(1,1)) , real(p(1,2)), imag(p(1,2)) ];
end
```

For *L'Hôspital's rule*, we differentiate the numerator and denominator of the function, $p(\lambda)$, to obtain, $P(\lambda)$, and use the latter instead, at the critical point, $\lambda = \sqrt{|c|/|a|}$, to obtain the correct value of the function there. We include the MATLAB code function[7], `psolqvh(r)`, shown in this section, which replaces the usual, `psolqv(r)`, to be used just at that critical point when desired. The function calculates the derivatives of the numerator and denominator, first, then computes the ratio, using the internal 2×2 complex number matrix representation of the global variables, a, b, c, then extracts the essential 4 components of from the result, and returns a quaternion in row vector format representing the provisional unit direction at the point where, $\lambda = r$. We append the letter '`h`' to the regular routine, `psolqv(r)`, that normally otherwise provides this function, to remind us that this is *L'Hôspital's* version of p.

Unfortunately, it is not always sufficient to differentiate just once, to find the finite ratio of the function. So, we need to modify the code above to include additional processing to complete the *L'Hôspital's rule*.

---

[7] All code appearing in this text is optimized for clarity, rather than being optimized for speed, memory resource usage, or floating point calculation error minimization. The reader may profitably re-write these routines to be more efficient in one of these other directions.

#2nd: we differentiate twice..

$$P_2(\lambda)=\frac{\left(\begin{array}{l}-16\lambda^9(a^*aa^*)(ba^*a)+52\lambda^7(a^*aa^*)(cb^*-\lambda^2ba^*)a-22\lambda^5(\lambda^4a^*aa^*-c^*ac^*)(ba^*a)\\+20\lambda^3(\lambda^4a^*aa^*-c^*ac^*)(cb^*-\lambda^2ba^*)a-20\lambda^3(a^*ca^*)(cb^*-\lambda^2ba^*)c\\+16\lambda^5(a^*ca^*)(ba^*c)-6\lambda(c^*cc^*-\lambda^4a^*ca^*)(ba^*c)\end{array}\right)}{\left(\begin{array}{l}(2\lambda^2|a|^2-(ca^*+ac^*))\cdot\left[\,(16\lambda^4|a|^2)(\lambda^4|a|^2+\lambda^2(ca^*+ac^*)+|c|^2)\,\right]\\+|\lambda^2a-c|^2\cdot\left[\,(4\lambda^2|a|^2)(11\lambda^4|a|^2+7\lambda^2(ca^*+ac^*)+3|c|^2)\,\right]\\+(\lambda^4|a|^2-|c|^2)\cdot\left[\begin{array}{r}(12\lambda^2|a|^2-2(ca^*+ac^*))(\lambda^4|a|^2+\lambda^2(ca^*+ac^*)+|c|^2)\\+(\lambda^4|a|^2-\lambda^2(ca^*+ac^*)+|c|^2)(12\lambda^2|a|^2+2(ca^*+ac^*))\\+2(4\lambda^3|a|^2-2\lambda(ca^*+ac^*))(4\lambda^3|a|^2+2\lambda(ca^*+ac^*))\end{array}\right]\end{array}\right)}\tag{1.637}$$

#3rd: we differentiate thrice..

$$P_3(\lambda)=\frac{\left(\begin{array}{r}-990\lambda^8\cdot(a^*aa^*\cdot ba^*a)+504\lambda^6\cdot(a^*aa^*\cdot cb^*a)+210\lambda^4\cdot(c^*ac^*\cdot ba^*a+a^*ca^*\cdot ba^*c)\\-60\lambda^2(c^*ac^*\cdot cb^*a+a^*ca^*\cdot cb^*c)-6(c^*cc^*\cdot ba^*c)\end{array}\right)}{\left(\begin{array}{r}+(96\lambda^3|a|^2)(3\lambda^2|a|^2-(ca^*+ac^*))|\lambda^2a+c|^2\\+(96\lambda^5|a|^2)(2\lambda^2|a|^2-(ca^*+ac^*))(2\lambda^2|a|^2+(ca^*+ac^*))\\+|\lambda^2a-c|^2(24\lambda|a|^2)(13\lambda^4|a|^2+5\lambda^2(ca^*+ac^*)+|c|^2)\\+(\lambda^4|a|^2-|c|^2)\cdot\left[\begin{array}{r}48\lambda|a|^2(\lambda^4|a|^2+|c|^2)\\+12\lambda(2\lambda^2|a|^2-(ca^*+ac^*))(6\lambda^2|a|^2+(ca^*+ac^*))\\+12\lambda(6\lambda^2|a|^2-(ca^*+ac^*))(2\lambda^2|a|^2+(ca^*+ac^*))\end{array}\right]\end{array}\right)}\tag{1.638}$$

```
function P=psolqvh(r) % ver:  1.0, (aqq + bq + c = 0)
global a b c;
ac = a'*c; acImagTest = abs(imag(ac(1,1))) + abs(real(ac(1,2))) + abs(imag(ac(1,2)));
if ~(acImagTest == 0 && real(ac(1,1)) > 0)
   % use p = P1
   np = 5*r^4*(r^4*a'*a*a' - c'*a*c')*(c*b' - r^2*b*a')*a;
   np = np + r^5*(4*r^3*a'*a*a')*(c*b' - r^2*b*a')*a;
   np = np + r^5*(r^4*a'*a*a' - c'*a*c')*(-2*r*b*a')*a;
   np = np + (c'*c*c' - r^4*a'*c*a')*(c*b' - r^2*b*a')*c;
   np = np + r*(-4*r^3*a'*c*a')*(c*b' - r^2*b*a')*c;
   np = np + r*(c'*c*c' - r^4*a'*c*a')*(- 2*r*b*a')*c;
   dp = trace((4*r^3*a*a')*(r^4*a*a' - r^2*(c*a' + a*c') + c*c')*(r^4*a*a' + r^2*(c*a' + a*c') + c*c'))/2;
   dp = dp + trace((r^4*a*a' - c*c')*(4*r^3*a*a' - 2*r*(c*a' + a*c'))*(r^4*a*a' + r^2*(c*a' + a*c') + c*c'))/2;
   dp = dp + trace((r^4*a*a' - c*c')*(r^4*a*a' - r^2*(c*a' + a*c') + c*c')*(4*r^3*a*a' + 2*r*(c*a' + a*c')))/2;
else   % use p = P3
   np = -990*r^8*(a'*a*a'*b*a'*a) + 504*r^6*(a'*a*a'*c*b'*a) + 210*r^4*(c'*a*c'*b*a'*a + a'*c*a'*b*a'*c);
   np = np - 60*r^2*(c'*a*c'*c*b'*a + a'*c*a'*c*b'*c) - 6*(c'*c*c'*b*a'*c);
   dp = trace(96*r^3*(a*a')*(3*r^2*a*a' - (c*a' + a*c'))*(r^2*a + c)*(r^2*a' + c'))/2;
   dp = dp + trace(96*r^5*(a*a')*(2*r^2*a*a' - (c*a' + a*c'))*(2*r^2*a*a' + (c*a' + a*c')))/2;
   dp = dp + trace((r^2*a - c)*(r^2*a' - c')*(24*r*(a*a'))*(13*r^4*a*a' + 5*r^2*(c*a' + a*c') + c*c'))/2;
   r4a2c2 = real(trace(r^4*a*a' - c*c')/2);
   if r4a2c2 ~= 0 % add (r^4.|a|^2 - |c|^2)...part!
    dp1 = trace(48*r*(a*a')*(r^4*a*a' + c*c'))/2;
    dp1 = dp1 + trace(12*r*(2*r^2*a*a' - (c*a' + a*c'))*(6*r^2*a*a' + (c*a' + a*c')))/2;
    dp1 = dp1 + trace(12*r*(6*r^2*a*a' - (c*a' + a*c'))*(2*r^2*a*a' + (c*a' + a*c')))/2;
    dp = dp + r4a2c2*dp1;
   end
end
p = np/dp;
P = [ real(p(1,1)), imag(p(1,1)), real(p(1,2)), imag(p(1,2)) ];
end
```

Let's now write, $P_1(\lambda) = P(\lambda)$, as the ratio of 1st derivatives, and compute, $P_2(\lambda)$, the ratio of 2nd derivatives, and also find, $P_3(\lambda)$, the ratio of 3rd derivatives.

When the two roots have the same magnitudes, then, $\lambda = \sqrt{|c|/|a|}$, and thus, $(\lambda^4|a|^2 - |c|^2) = 0$, and the formula for $P_1(\lambda)$ simplifies further, since the last two terms in its denominator drop away. However, if $a^*c \in \mathbb{R}, a^*c > 0$, then, $|\lambda^2 a - c|^2 = (\lambda^4 - \lambda^2(ca^* + ac^*) + |c|^2) = 0$, so the entire denominator vanishes, and then we can't use $P_1(\lambda)$. We must differentiate again, which gives us $P_2(\lambda)$. But, this denominator also vanishes under these conditions, so we need yet another turn at the derivatives, needing to calculate $P_3(\lambda)$. The formulas, and completed MATLAB code for the *L'Hôspital's* calculation, `psolqvh(r)`, is given above.

One advantage of the code shown above, is that we see clearly how it interfaces with the original parameters of the given quadratic equation, $aq^2 + bq + c = 0$, and working with these original parameters has the additional benefit that the numerator calculations on whole numbers sometimes produce a more accurate result, often enabling us to obtain actual 0's, instead of just small numbers, like, 10^{-15}. However, these are messy expressions with unwieldly formulas, and so we'll work out much more computationally efficient alternative formulas below. These higher derivatives are needed when, $b \neq 0$, while, $a^*c \in \mathbb{R},\ a^*c > 0$.

First, we're going to transform the three variables, (a, b, c), by dividing the quadratic equation throughout by the lead, a. Thus, we effectively set, $a = 1,\ b = a^{-1}b,\ c = a^{-1}c$. This division occasionally makes the calculation a bit more inaccurate, in numerical calculations, especially when given whole number coefficients for the parameters, even though algebraically things are identical in the transformed formulas. There are several alternative ways we can express the formulas for the quadratic equation solutions, and while all of them are algebraically identical, in numerical calculations they generally produce different results, owing to variation in round-off error accumulation. This is especially significant when working very near, or at the critical point. Away from the critical point, the variation in results from alternative formulas is less significant. We're also going to do some formula optimization, and code optimization, which again changes the numbers. From experience, it is found best not to try to crush the spike of a hybrid with an optimized formula. Remember, *L'Hôspital's Rule* is only really valid when we have a true zero in both numerator and denominator. Crushing a spike on a hybrid is technically illegal. But, it works reasonably well with the full formulas, but not generally the optimized versions of them. So, keep this in mind, when applying the following results to calculations. Let's consider our previously obtained optimized formula for $p(\lambda)$.

$$p(\lambda) = \frac{\lambda(\lambda^2 - c^*)(cb^*(\lambda^2 - c) - \lambda^2(\lambda^2 - c)b)}{(\lambda^4 - |c|^2)|\lambda^2 - c|^2} \tag{1.639}$$

We expand its numerator and denominator in prepration for differentiating.

$$P_0(\lambda) = \frac{(\lambda^5 cb^* - \lambda^3 c(b^*c + c^*b^*) + \lambda|c|^2 b^*c) - (\lambda^7 - \lambda^5(c + c^*) + \lambda^3|c|^2)b}{(\lambda^4 - |c|^2)(\lambda^4 - \lambda^2(c + c^*) + |c|^2)} \tag{1.640}$$

Now we compute three levels of derivatives.

#1st: we differentiate once...

$$P_1(\lambda) = \left[\frac{(5\lambda^4 cb^* - 3\lambda^2 c(b^*c + c^*b^*) + |c|^2 b^*c) - (7\lambda^6 - 5\lambda^4(c + c^*) + 3\lambda^2|c|^2)b}{(4\lambda^3)(\lambda^4 - \lambda^2(c + c^*) + |c|^2) + (\lambda^4 - |c|^2)(4\lambda^3 - 2\lambda(c + c^*))} \right] \tag{1.641}$$

#2nd: we differentiate twice...

$$P_2(\lambda) = \left[\frac{(20\lambda^3 cb^* - 6\lambda c(b^*c + c^*b^*)) - (42\lambda^5 - 20\lambda^3(c + c^*) + 6\lambda|c|^2)b}{(12\lambda^2)(\lambda^4 - \lambda^2(c + c^*) + |c|^2) + (8\lambda^3)(4\lambda^3 - 2\lambda(c + c^*)) + (\lambda^4 - |c|^2)(12\lambda^2 - 2(c + c^*))} \right] \tag{1.642}$$

#3rd: we differentiate thrice...

$$P_3(\lambda) = \left[\frac{(10\lambda^2 cb^* - c(b^*c + c^*b^*)) - (35\lambda^4 - 10\lambda^2(c + c^*) + |c|^2)b}{4\lambda^3(14\lambda^2 - 5(c + c^*))} \right] \tag{1.643}$$

Of course, we only need to evaluate this function at one point, $\lambda = \sqrt{|c|/|a|}$. But, here, with our re-scaling of parameters, this point is just, $\lambda = \sqrt{|c|}$. If we put this value of λ into the original formula, which we've relabeled, $P_0(\lambda)$, the denominator vanishes. So, we can't use that. What about the ratio of 1st derivatives, $P_1(\lambda)$? Let's see,

$$\#1\text{st:} \quad \lambda = \sqrt{|c|} \tag{1.644}$$
$$(4\lambda^3)(\lambda^4 - \lambda^2(c+c^*) + |c|^2) + (\lambda^4 - |c|^2)(4\lambda^3 - 2\lambda(c+c^*)) \tag{1.645}$$
$$= (4|c|^{3/2})(|c|^2 - |c|(c+c^*) + |c|^2) + (|c|^2 - |c|^2)(4|c|^{3/2} - 2|c|^{1/2}(c+c^*)) \tag{1.646}$$
$$= 4|c|^{5/2}(2|c| - (c+c^*)) \tag{1.647}$$
$$= 8|c|^{5/2}(|c| - S(c)) \tag{1.648}$$

So, when, $c \in \mathbb{R},\ c > 0$, then, $(|c| - S(c)) = 0$, and the ratio of 1st derivatives can't find the finite value of the function. This corresponds to the case, $a^*c \in \mathbb{R},\ a^*c > 0$, in the un-transformed variables. Other than this exception, the ratio of 1st derivatives works just fine. Since, for positive real value c parameters, the 1st is insufficient, we try the 2nd ratio; i.e., the ratio of second order derivatives.

$$\#2\text{nd:} \quad \lambda = \sqrt{|c|} \tag{1.649}$$
$$(12\lambda^2)(\lambda^4 - \lambda^2(c+c^*) + |c|^2) + (8\lambda^3)(4\lambda^3 - 2\lambda(c+c^*)) + (\lambda^4 - |c|^2)(12\lambda^2 - 2(c+c^*)) \tag{1.650}$$
$$= (12|c|)(|c|^2 - |c|(c+c^*) + |c|^2) + (8|c|^{3/2})(4|c|^{3/2} - 2|c|^{1/2}(c+c^*)) + (|c|^2 - |c|^2)(12|c| - 2(c+c^*)) \tag{1.651}$$
$$= 28|c|^2(2|c| - (c+c^*)) \tag{1.652}$$
$$= 56|c|^2(|c| - S(c)) \tag{1.653}$$

Again, however, we meet with the very same condition. Can't use the ratio of second order derivatives either. So, we try the 3rd option; the ratio of third order derivatives.

$$\#3\text{rd:} \quad \lambda = \sqrt{|c|} \tag{1.654}$$
$$4\lambda^3(14\lambda^2 - 5(c+c^*)) \tag{1.655}$$
$$= 4|c|^{3/2}(14|c| - 5(c+c^*)) \tag{1.656}$$
$$= 8|c|^{3/2}(7|c| - 5S(c)) \tag{1.657}$$

Finally, we have a non-zero value for the denominator. This means, we're either using, $P_1(\lambda = \sqrt{|c|})$, or, we're using, $P_3(\lambda = \sqrt{|c|})$, we never use the 2nd option; it's 1st order derivatives, or 3rd order derivatives, to apply *L'Hôspital's Rule* here. When, $P_1(\lambda)$, works, we have;

$$P_1(\lambda = \sqrt{|c|}) = \left[\frac{(5|c|^2cb^* - 3|c|c(b^*c + c^*b^*) + |c|^2b^*c) - (7|c|^3 - 5|c|^2(c+c^*) + 3|c||c|^2)b}{4|c|^{5/2}(2|c| - (c+c^*))} \right] \tag{1.658}$$

$$= \left[\frac{|c|^2 \cdot (\,5c - 3|c|\,) \cdot b^* \;-\; |c| \cdot (\,3c - |c|\,) \cdot b^*c \;-\; 5|c|^2 \cdot (\,2|c| - (c+c^*)\,) \cdot b}{4|c|^{5/2}(2|c| - (c+c^*))} \right] \tag{1.659}$$

When we have to use, $P_3(\lambda)$, the result is,

$$P_3(\lambda = \sqrt{|c|}) = \left[\frac{(60|c|cb^* - 6c(b^*c + c^*b^*)) - (210|c|^2 - 60|c|(c+c^*) + 6|c|^2)b}{24|c|^{3/2} \cdot [\,14|c| - 5(c+c^*)\,]} \right] \tag{1.660}$$

$$= \left[\frac{(60|c|cb^* - 6c(b^*c + c^*b^*)) - (216|c|^2 - 60|c|(c+c^*))b}{24|c|^{3/2} \cdot [\,14|c| - 5(c+c^*)\,]} \right] \tag{1.661}$$

$$= \left[\frac{[\,10|c|cb^* - c(b^*c + c^*b^*)\,] \;-\; [\,36|c|^2 - 10|c|(c+c^*)\,]b}{4|c|^{3/2} \cdot [\,14|c| - 5(c+c^*)\,]} \right] \tag{1.662}$$

We can simplify these two expressions further to reduce the number of arithmetic operations required to find the two, $P_1(\lambda = \sqrt{|c|})$, and, $P_3(\lambda = \sqrt{|c|})$, variants of *L'Hôspital's* calculation.

First consider, P_1, we can re-write;

$$P_1(\lambda = \sqrt{|c|}) \quad = \frac{1}{\sqrt{|c|}} \cdot \left[\frac{|c|^2 \cdot (\, 5c - 3|c| \,) \cdot b^* \;\; - \;\; |c| \cdot (\, 3c - |c| \,) \cdot b^* c}{4|c|^2(2|c| - (c + c^*))} - \frac{5b}{4} \right] \tag{1.663}$$

so, let's see if we can simplify the numerator some more,

$$|c|^2(5c - 3|c|)b^* - |c|(3c - |c|)b^* c \tag{1.664}$$

$$= |c|^2(5c - 3|c|)b^* - |c|(3c - |c|)(cb^* + [b^*, c]) \tag{1.665}$$

$$= |c|^2(5c - 3|c|)b^* - |c|(3c - |c|)(cb^* - [b, c]) \tag{1.666}$$

$$= |c|^2(5c - 3|c|)b^* - |c|(3c - |c|)cb^* + |c|(3c - |c|)[b, c] \tag{1.667}$$

$$= (|c|^2(5c - 3|c|) - |c|(3c - |c|)c)b^* + |c|(3c - |c|)[b, c] \tag{1.668}$$

$$= |c|(6|c| - 3c^* - 3c)cb^* + |c|(3c - |c|)[b, c] \tag{1.669}$$

$$= (2|c| - (c + c^*)) \cdot 3|c|cb^* + |c|(3c - |c|)[b, c] \tag{1.670}$$

we can then have;

$$P_1(\lambda = \sqrt{|c|}) \quad = \frac{1}{4\sqrt{|c|}} \cdot \left[\frac{(3c - |c|)[b, c]}{|c|(2|c| - (c + c^*))} + \frac{3cb^*}{|c|} - 5b \right] \tag{1.671}$$

It isn't clear that we gain any advantage here, so we'll just stick with the previous result, and write, with a bit of rearranging,

$$P_1(\lambda = \sqrt{|c|}) \quad = \frac{1}{\sqrt{|c|}} \cdot \frac{1}{4} \left[\frac{(\, 5c - 3|c| \,)b^* \;\; + \;\; (\, 1 - 3c/|c| \,)b^* c}{2|c| - (c + c^*)} \;\; - \;\; 5b \right] \tag{1.672}$$

and we'll adopt this for our official expression for P_1. Next, we take a look at P_3, where we recognise the fact that we only need to call P_3, when the condition, $2|c| - (c + c^*) = 0$, holds, and also, $c \in \mathbb{R},\ c > 0$. Remember, though, these are the "*lead reduced*" parameters. So, really, it's, $a^* c \in \mathbb{R},\ a^* c > 0$. This means we can use the identity, $2|c| = c + c^*$, and, $|c| = c = c^*$, here, to reduce the expression further. So,

$$P_3(\lambda = \sqrt{|c|}) \quad = \left[\frac{[\, 10|c|cb^* - c(b^* c + c^* b^*) \,] \;\; - \;\; [\, 36|c|^2 - 10|c|(c + c^*) \,]b}{4|c|^{3/2} \cdot [\, 14|c| - 5(c + c^*) \,]} \right] \tag{1.673}$$

$$= \left[\frac{[\, 10|c||c|b^* - |c|(b^*|c| + |c|b^*) \,] \;\; - \;\; [\, 36|c|^2 - 10|c|(2|c|) \,]b}{4|c|^{3/2} \cdot [\, 14|c| - 10|c| \,]} \right] \tag{1.674}$$

$$= \frac{1}{\sqrt{|c|}} \cdot \left[\frac{[\, 8|c|^2 \,]b^* \;\; - \;\; [\, 16|c|^2 \,]b}{16|c|^2} \right] \tag{1.675}$$

$$= \frac{1}{\sqrt{|c|}} \cdot \left[\frac{1}{2} b^* - b \right] \tag{1.676}$$

Now we'd like to transform back to our original formal parameters, (a, b, c), to see these 1st and 3rd order results in terms of parameters of the given equation. Putting back in, $c \to a^{-1}c$, and, $b \to a^{-1}b$, we obtain,

#1st:

$$P_1(\lambda = \sqrt{|c|}) \;\; = \frac{1}{\sqrt{|c|}} \cdot \frac{1}{4} \left[\frac{(\,5c - 3|c|\,)b^* \;\; + \;\; (\,1 - 3c/|c|\,)b^*c}{2|c| - (c + c^*)} \;\; - \;\; 5b \right] \tag{1.677}$$

$c \to a^{-1}c,\; b \to a^{-1}b$:

$$\therefore \quad P_1(\lambda = \sqrt{|c|/|a|}) \;\; = \frac{1}{\sqrt{|c|/|a|}} \cdot \frac{1}{4} \left[\frac{(\,5a^*c - 3|a^*c|\,)b^*a \;\; + \;\; |a|^2(\,1 - 3a^*c/|a^*c|\,)b^*c}{|a|^2(\,2|a^*c| - (a^*c + (a^*c)^*)\,)} \;\; - \;\; \frac{5a^*b}{|a|^2} \right] \tag{1.678}$$

#3rd:

$$P_3(\lambda = \sqrt{|c|}) \;\; = \frac{1}{\sqrt{|c|}} \cdot \left[\frac{1}{2}b^* - b \right] \tag{1.679}$$

$c \to a^{-1}c,\; b \to a^{-1}b$:

$$\therefore \quad P_3(\lambda = \sqrt{|c|/|a|}) \;\; = \frac{1}{\sqrt{|c|/|a|}} \cdot \left[\frac{b^*a - 2a^*b}{2|a|^2} \right] \tag{1.680}$$

These are then the final forms of the *L'Hôspital's Rule* formulas that we need for the work.

Of course, it is not necessary to provide a whole range function, like, `psolqvh(r)`, that produces a range of values, for input domain, $\lambda = r \in (-\infty, +\infty)$, since we only need to know one number, which is the evaluation at a particular point, $\lambda = \sqrt{|c|/a|}$. But, sometimes it's useful when exploring things, to investigate the neighborhood of the point.

Therefore, we provide a simpler MATLAB function, called `psolqvh0`, that takes no arguments, and that just returns the value of $p(\lambda = \sqrt{|c|/|a|})$, at that one point itself.

in MATLAB code:

```
function P=psolqvh0 % ver:  1.0, (aqq + bq + c = 0)
global a b c;
ac = a'*c;
acImagTest = abs(imag(ac(1,1))) + abs(real(ac(1,2))) + abs(imag(ac(1,2)));
r = sqrt(sqrt(trace(c*c')/trace(a*a')));
I = [1,0;0,1];
mac = sqrt(real(trace(ac*ac')/2)); % = |a*.c|
if acImagTest == 0 && real(ac(1,1)) > 0 % a*.c in [R], a*.c > 0
  % use P = P3
  p = (b'*a - 2*a'*b)/2;
else
  % use P = P1
  p = ((5*a'*c - 3*mac*I)*b'*a + (a*a')*(I - 3*a'*c/mac)*b'*c)/(2*mac - 2*real(ac(1,1)));
  p = ( p - 5*a'*b )/4;
end
p = (1/r)*p/real(trace(a*a')/2);
P = [ real(p(1,1)), imag(p(1,1)), real(p(1,2)), imag(p(1,2)) ];
end
```

We use this last function, `psolqvh0`, in the final source code for the function, `[Q1 Q2] = aqqbqc(A,B,C)`, listed in the "*reference*" section below, once things are complete and understood. But, otherwise within the text, while exploring quadratic solutions, we prefer to use, `psolqvh(r)`, when investigating and discussing the quadratic equations. While both, `psolqvh(r)` and `psolqvh0`, generally give identical results when *L'Hôspital's* conditions

are satisfied, they differ significantly otherwise when deliberately re-interpreting data that doesn't exactly fit those exacting specifications; such as when crushing the spike on a hybrid.

in MATLAB code:

```
function P=psolqvh0 % ver:  1.1, (aqq + bq + c = 0)
global a b c;
cc = c; bb = b; % save global vars
I = [1,0;0,1]; c = (a^-1)*c; b = (a^-1)*b; % rescale
cImagTest = abs(imag(c(1,1))) + abs(real(c(1,2))) + abs(imag(c(1,2)));
mc = sqrt(real(trace(c*c')/2)); % = |c|
r = sqrt(mc); % = rt|c|
if cImagTest == 0 && real(c(1,1)) > 0 % a*.c in [R], a*.c > 0
  % use P = P3
  p = b'/2 - b;
else
  % use P = P1
  p = (((5*c - 3*mc*I)*b' + (I - 3*c/mc)*b'*c)/(2*mc - 2*real(c(1,1))) - 5*b)/4;
end
p = (1/r)*p;
P = [ real(p(1,1)), imag(p(1,1)), real(p(1,2)), imag(p(1,2)) ];
c = cc; b = bb; % restore global vars
end
```

Finally, we illustrate an alternative implementation, `P=psolqvh0  ver:  1.1`, that uses the "lead reduced" method, requiring the "save and restore" of global vars, that does the same job.

OPTIMIZING FORMULAE AND CODE:

There are various ways to re-arrange the standard formula for the provisional unit direction function, e.g.,

$$p(\lambda) \quad = \quad \frac{\lambda^5(\lambda^4 a^* a a^* - c^* a c^*)(cb^* - \lambda^2 b a^*)a + \lambda(c^* c c^* - \lambda^4 a^* c a^*)(cb^* - \lambda^2 b a^*)c}{(\lambda^4|a|^2 - |c|^2)\cdot|\lambda^2 a - c|^2 \cdot |\lambda^2 a + c|^2} \tag{1.681}$$

we've seen the form obtained, dividing by the lead parameter, a, thus replacing, $c = a^{-1}c, b = a^{-1}b$, yielding the alternate form,

$$p(\lambda) = \frac{\lambda(\lambda^2 - c^*)(cb^*(\lambda^2 - c) - \lambda^2(\lambda^2 - c)b)}{(\lambda^4 - |c|^2)|\lambda^2 - c|^2} \tag{1.682}$$

then, if we put back in the lead parameter, a, we obtain the shorter version, in full regalia, (a, b, c), viz;

$$p(\lambda) = \frac{\lambda(\lambda^2 a^* - c^*)(cb^*(\lambda^2 a - c) - \lambda^2(\lambda^2 a - c)a^* b)}{(\lambda^4|a|^2 - |c|^2)|\lambda^2 a - c|^2} \tag{1.683}$$

which we may re-arrange yet again to,

$$p(\lambda) = \frac{\lambda(\lambda^2 a^* - c^*)cb^*(\lambda^2 a^* - c^*)^{-1} - \lambda^3 a^* b}{(\lambda^4|a|^2 - |c|^2)} \tag{1.684}$$

implementing...

```
  in MATLAB code:

function p=psolqd(r)    % ver:  1.1,    (aqq + bq + c = 0)
global a b c;
p = (r*(r^2*a' - c')*c*b'*(r^2*a' - c')^-1 - r^3*a'*b)/(trace(r^4*a*a' - c*c')/2);
end
```

thus obtaining a version that takes just one line of code. We could again reduce this, once more setting, $a = 1$, with the usual, $c = a^{-1}c, b = a^{-1}b$, to get,

$$p(\lambda) = \frac{\lambda(\lambda^2 - c^*)cb^*(\lambda^2 - c^*)^{-1} - \lambda^3 b}{(\lambda^4 - |c|^2)} \tag{1.685}$$

implementing...

```
  in MATLAB code:

function p=psolqd(r)    % ver:  1.2,    (aqq + bq + c = 0)
global a b c;
cc = c; bb = b; % save global vars
I = [1,0;0,1];
c = (a^-1)*c; b = (a^-1)*b;
p = (r*(r^2*I - c')*c*b'*(r^2*I - c')^-1 - r^3*b)/(trace(r^4*I - c*c')/2);
c = cc; b = bb; % restore global vars
end
```

but must remember to transform the original vars, then save and restore the globals, and not forget to include the unit matrix, $I = [1, 0; 0, 1]$, to implement the code. Also recall, taking the "inverse" of a matrix is generally a more computationally expensive operation than just multiplication, so formulas that look shorter may not be computationally more efficient. Thus there are several alternate ways to implement the solutions. Each method has its own merits.

`Optimal Magnitudes?:` Is there an optimal way to find the λ magnitudes? Consider the 1st optimized form for $p(\lambda)$, that is already "lead reduced," by dividing throughout by the 'a' parameter,

$$p(\lambda) = \frac{\lambda(\lambda^2 - c^*)(cb^*(\lambda^2 - c) - \lambda^2(\lambda^2 - c)b)}{(\lambda^4 - |c|^2)|\lambda^2 - c|^2} \tag{1.686}$$

Let's use this to compute and set the usual constraint, $|p|^2 = p^*p = 1$,

$$\begin{aligned}
&[\,\lambda(\lambda^2 - c^*)(cb^*(\lambda^2 - c) - \lambda^2(\lambda^2 - c)b)\,]^* \cdot [\,\lambda(\lambda^2 - c^*)(cb^*(\lambda^2 - c) - \lambda^2(\lambda^2 - c)b)\,] \\
&= (\lambda^4 - |c|^2)^2 \cdot |\lambda^2 - c|^4 && (1.687)\\
&\text{numerator:} \\
&= [\,\lambda((\lambda^2 - c^*)bc^* - \lambda^2 b^*(\lambda^2 - c^*))(\lambda^2 - c)\,] \cdot [\,\lambda(\lambda^2 - c^*)(cb^*(\lambda^2 - c) - \lambda^2(\lambda^2 - c)b)\,] && (1.688)\\
&= \lambda^2 \cdot |\lambda^2 - c|^2 \cdot ((\lambda^2 - c^*)bc^* - \lambda^2 b^*(\lambda^2 - c^*)) \cdot (cb^*(\lambda^2 - c) - \lambda^2(\lambda^2 - c)b) && (1.689)\\
&\text{removing one } |\lambda^2 - c|^2 \text{ factor from both sides:} \\
&\lambda^2 \cdot ((\lambda^2 - c^*)bc^* - \lambda^2 b^*(\lambda^2 - c^*)) \cdot (cb^*(\lambda^2 - c) - \lambda^2(\lambda^2 - c)b) \;=\; (\lambda^4 - |c|^2)^2 \cdot |\lambda^2 - c|^2 && (1.690)\\
&= \lambda^2 \cdot (|b|^2|c|^2|\lambda^2 - c|^2 - \lambda^2(\lambda^2 - c^*)bc^*(\lambda^2 - c)b - \lambda^2 b^*(\lambda^2 - c^*)cb^*(\lambda^2 - c) + \lambda^4|b|^2|(\lambda^2 - c)|^2) && (1.691)
\end{aligned}$$

Now we permute parameters using the various conjugation and commutator bracket rules:

$$[b, c]^* = [b, c^*] = [b^*, c] = -[b, c], \qquad \text{and,} \qquad [b, c]c^* = c[b, c],\ [b, c]c = c^*[b, c], \qquad \text{etc..} \tag{1.692}$$

$$
\begin{aligned}
&(\lambda^2 - c^*)bc^*(\lambda^2 - c)b && + \quad b^*(\lambda^2 - c^*)cb^*(\lambda^2 - c) && (1.693)\\
&= (\lambda^2 b - c^* b)c^*(\lambda^2 b - cb) && + \quad b^*(\lambda^2 c - c^* c)b^*(\lambda^2 - c) && (1.694)\\
&= (\lambda^2 b - bc^* - [c^*, b])c^*(\lambda^2 b - bc - [c, b]) && + \quad b^* c(\lambda^2 b^* - c^* b^*)(\lambda^2 - c) && (1.695)\\
&= (b(\lambda^2 - c^*)c^* - [b, c]c^*)(b(\lambda^2 - c) + [b, c]) && + \quad b^* c(\lambda^2 b^* - b^* c^* - [c^*, b^*])(\lambda^2 - c) && (1.696)\\
&= (bc^*(\lambda^2 - c^*) - c[b, c])(b(\lambda^2 - c) + [b, c]) && + \quad b^* c(b^*(\lambda^2 - c^*) + [b, c])(\lambda^2 - c) && (1.697)\\
&= bc^*(\lambda^2 - c^*)b(\lambda^2 - c) + bc^*(\lambda^2 - c^*)[b, c] - c[b, c]b(\lambda^2 - c) - c[b, c]^2 && + \quad b^* cb^*(\lambda^2 - c^*)(\lambda^2 - c) + b^* c[b, c](\lambda^2 - c) && (1.698)\\
&= bc^*(\lambda^2 b - c^* b)(\lambda^2 - c) + bc^*[b, c](\lambda^2 - c) - c[b, c]b(\lambda^2 - c) - c[b, c]^2 && + \quad b^* cb^*|\lambda^2 - c|^2 + b^* c[b, c](\lambda^2 - c) && (1.699)\\
&= bc^*(\lambda^2 b - bc^* - [c^*, b])(\lambda^2 - c) + (bc^* - cb^*)[b, c](\lambda^2 - c) - c[b, c]^2 && + \quad b^* cb^*|\lambda^2 - c|^2 + b^* c[b, c](\lambda^2 - c) && (1.700)\\
&= bc^*(b(\lambda^2 - c^*) - [b, c])(\lambda^2 - c) + (bc^* - cb^*)[b, c](\lambda^2 - c) - c[b, c]^2 && + \quad b^* cb^*|\lambda^2 - c|^2 + b^* c[b, c](\lambda^2 - c) && (1.701)\\
&= bc^* b|\lambda^2 - c|^2 - bc^*[b, c](\lambda^2 - c) + (bc^* - cb^*)[b, c](\lambda^2 - c) - c[b, c]^2 && + \quad b^* cb^*|\lambda^2 - c|^2 + b^* c[b, c](\lambda^2 - c) && (1.702)\\
&= (bc^* b + b^* cb^*)|\lambda^2 - c|^2 + (b^* c - cb^*)[b, c](\lambda^2 - c) - c[b, c]^2 && && (1.703)\\
&= (bc^* b + b^* cb^*)|\lambda^2 - c|^2 + [b^*, c][b, c](\lambda^2 - c) - c[b, c]^2 && && (1.704)\\
&= (bc^* b + b^* cb^*)|\lambda^2 - c|^2 - [b, c][b, c](\lambda^2 - c) - c[b, c]^2 && && (1.705)\\
&= (bc^* b + b^* cb^*)|\lambda^2 - c|^2 - [b, c]^2\lambda^2 && && (1.706)
\end{aligned}
$$

What we have then, is,

$$\lambda^2 \cdot [\, |b|^2|c|^2|\lambda^2 - c|^2 - \lambda^2(bc^* b + b^* cb^*)|\lambda^2 - c|^2 + [b, c]^2\lambda^4 + \lambda^4|b|^2|\lambda^2 - c|^2 \,] \;=\; (\lambda^4 - |c|^2)^2 \cdot |\lambda^2 - c|^2 \quad (1.707)$$

$$\lambda^2 \cdot [\, \lambda^4|b|^2 - \lambda^2(bc^* b + b^* cb^*) + |b|^2|c|^2 \,] \cdot |\lambda^2 - c|^2 + \lambda^6 \cdot [b, c]^2 \;=\; (\lambda^4 - |c|^2)^2 \cdot |\lambda^2 - c|^2 \quad (1.708)$$

re-arranging,

$$\big[\, (\lambda^4 - |c|^2)^2 - \lambda^2 \cdot [\, \lambda^4|b|^2 - \lambda^2(bc^* b + b^* cb^*) + |b|^2|c|^2 \,] \,\big] \cdot |\lambda^2 - c|^2 \;=\; \lambda^6 \cdot [b, c]^2 \quad (1.709)$$

$$\big[\, (\lambda^2)^4 \;-\; (\lambda^2)^3|b|^2 \;+\; (\lambda^2)^2(-2|c|^2 + (bc^* b + b^* cb^*)) \;-\; (\lambda^2)|b|^2|c|^2 \;+\; |c|^4 \,\big] \cdot |\lambda^2 - c|^2 \;=\; \lambda^6 \cdot [b, c]^2 \quad (1.710)$$

We see that when the "lead reduced" parameters commute, $[b, c] = 0$, then the polynomial reduces to a factor expression containing a 4th order polynomial in the square magnitude, λ^2, as the highest degree equation required to solve for the magnitude. Thus, in that case, we can solve using the well known quartic solution, obtaining results for λ in the form of the usual radical expressions. And since, λ, is the only thing we have to solve for numerically, in the *two-step* method, the complete solution, $q_1 = \lambda_1 \cdot p_1$, and, $q_2 = \lambda_2 \cdot p_2$, is determined by the usual closed form radical expressions under this situation. We code-up a MATLAB function to construct the coefficients of this 4th degree equation, `v=pvec4`, conveniently using our global variables, and 2×2 complex matrix quaternion representation, to calculate with, as shown below;

in MATLAB code:

```
function v=pvec4 % ver:  1.0, (aqq + bq + c = 0)
global a b c;
cc = c; bb = b; % save global vars
c = (a^-1)*c; b = (a^-1)*b; % rescale
% compute poly coefs
a3 = -b*b';
a2 = -2*c*c' + (b*c'*b + b'*c*b');
a1 = -(b*b')*(c*c');
a0 = (c*c')*(c*c');
% extract scalar vals from 2x2 matrices
s3 = real(a3(1,1));
s2 = real(a2(1,1));
s1 = real(a1(1,1));
s0 = real(a0(1,1));
c = cc; b = bb; % restore global vars
v = [1,s3,s2,s1,s0];
end
```

We can check this "4th" order polynomial method out, by re-exploring our `onplane` example from above, since all parameters, $(a, b, c) \in \mathbb{C} \subset \mathbb{H}$, obey the commutation, so that, $[b, c] = 0$, for lead reduced parameters, in this case. Using MATLAB's built-in "`roots()`" function, we see the two real roots, $\lambda_1 = \lambda_2 = 5$, as before.

```
>> global a b c
>> A = [1,0,0,0]                                A = 1 0 0 0
>> B = [-112,6,-4,12]/13                        B = -8.6154 0.4615 -0.3077 0.9231
>> C = [2205,-240,160,-480]/91                  C = 24.2308 -2.6374 1.7582 -5.2747
>> psolqabc(A,B,C)
>> r = sqrt(sqrt(trace(c*c')/trace(a*a')))      r = 5
>> v = pvec4                                     v = 1.0e+05 * 0.0000 -0.0008 0.0252 -0.4712 3.9063
>> rt = sqrt(roots(v))
rt =
4.3412 + 2.4807i
4.3412 - 2.4807i
5.0000 + 0.0000i
5.0000 - 0.0000i
>>
```

However, more generally, $[b, c] \neq 0$, and, when using this approach, we have to solve a higher "6th" degree polynomial to obtain λ^2, hence, λ, with the details shown below;

$$\lambda^{2\cdot 6} + \alpha_5\lambda^{2\cdot 5} + \alpha_4\lambda^{2\cdot 4} + \alpha_3\lambda^{2\cdot 3} + \alpha_2\lambda^{2\cdot 2} + \alpha_1\lambda^{2\cdot 1} + \alpha_0 = 0 \tag{1.711}$$

$$\begin{aligned}
\alpha_5 &= -|b|^2 - (c + c^*) \\
\alpha_4 &= -|c|^2 + |b|^2(c + c^*) + (bc^*b + b^*cb^*) \\
\alpha_3 &= -2|c|^2(|b|^2 - (c + c^*)) - (bc^*b + b^*cb^*)(c + c^*) - [b, c]^2 \\
\alpha_2 &= [-|c|^2 + |b|^2(c + c^*) + (bc^*b + b^*cb^*)] \cdot |c|^2 \\
\alpha_1 &= [-|b|^2 - (c + c^*)] \cdot |c|^4 \\
\alpha_0 &= +|c|^6
\end{aligned} \tag{1.712}$$

The coefficients are built from the five scalars, $|b|^2,\ |c|^2,\ (c + c^*),\ (bc^*b + b^*cb^*),\ [b, c]^2,$ their powers and products.

in MATLAB code:

```
function v=pvec6 % ver:  1.0, (aqq + bq + c = 0)
global a b c;
cc = c; bb = b; % save global vars
c = (a^-1)*c; b = (a^-1)*b; % rescale
% compute poly coefs
a5 = -b*b' - (c + c');
a4 = -c*c' + (b*b')*(c + c') + (b*c'*b + b'*c*b');
a3 = -2*(c*c')*(b*b' - (c + c')) - (b*c'*b + b'*c*b')*(c + c') - (b*c - c*b)^2;
a2 = (-(c*c') + (b*b')*(c + c') + (b*c'*b + b'*c*b'))*(c*c');
a1 = (-(b*b') - (c + c'))*(c*c')*(c*c');
a0 = (c*c')*(c*c')*(c*c');
% extract scalar vals from 2x2 matrices
s5 = real(a5(1,1));
s4 = real(a4(1,1));
s3 = real(a3(1,1));
s2 = real(a2(1,1));
s1 = real(a1(1,1));
s0 = real(a0(1,1));
c = cc; b = bb; % restore global vars
v = [1,s5,s4,s3,s2,s1,s0];
end
```

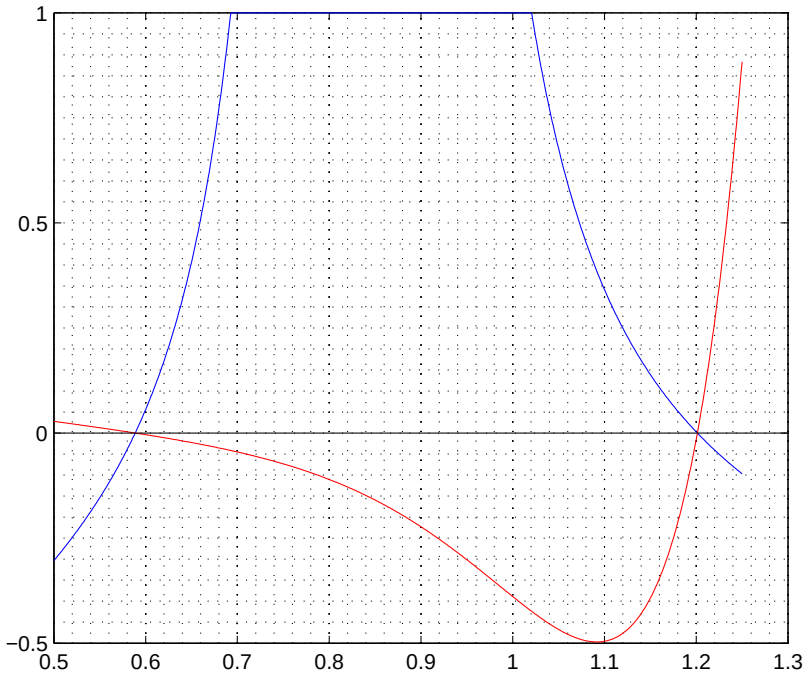


FIG. 11: Plot of psolqcap(r) , $f(\lambda) = p(\lambda)p(\lambda)^* - 1$; (blue) $aq^2 + bq + c = 0$, and, corresponding characteristic 6th degree polynomial function, $f(\lambda) = \lambda^{2\cdot 6} + \alpha_5\lambda^{2\cdot 5} + \alpha_4\lambda^{2\cdot 4} + \alpha_3\lambda^{2\cdot 3} + \alpha_2\lambda^{2\cdot 2} + \alpha_1\lambda^{2\cdot 1} + \alpha_0$, (red), showing the two roots, $\lambda_1 = 0.5886$, $\lambda_2 = 1.2013$, for the quaternion quadratic with parameters, $a = 2 + i - k$, $b = -3 + j + k$, $c = 1 - j + k$.

```
  .
>> global a b c
>> A = [2, 1, 0, -1 ];
>> B = [-3, 0, 1, 1 ];
>> C = [ 1, 0,-1, 1 ];
>> psolqabc(A,B,C);
>> v = pvec6
v =
1.0000 -2.1667 1.8333 -1.5556 0.9167 -0.5417 0.1250
>> rt = sqrt(roots(v))
rt =
1.2013 + 0.0000i
0.4972 + 0.6782i
0.4972 - 0.6782i
0.7444 + 0.3910i
0.7444 - 0.3910i
0.5886 + 0.0000i
>>
>> vv = [v(1),0,v(2),0,v(3),0,v(4),0,v(5),0,v(6),0,v(7)];
>> clear rr yy
>> rr = 0.5:0.001:1.25;
>> zz = polyval(vv,rr);
>> size(rr)
ans =
1 751
>> for i = 1:751
yy(i)=psolqcap(rr(i));
```

```
end
>> plot(rr,yy,'b',rr,zz,'r')
>> hold on
>> plot([.5 1.3],[0 0],'k-')
>> grid on
>> grid minor
>>
```

Unlike the provisional unit curve, which has a natural spike, marking the spot where we should look for the roots, the 6th degree polynomial curve is characteristically unrevealing by contrast, and unless you know where to look for the roots, it's difficult to see them in the average plot. This is because the polynomial curve blows up generally, as $\lambda \to +\infty$, while the $p(\lambda)$ function does the opposite, going to 0 at the ends of the range, so the $p(\lambda)$ is a better fit for the search for solutions. We could plot the inverse, $1/f(\lambda)$, for the polynomial, but then often need to also use a finer grid of plot points to find the root locations.

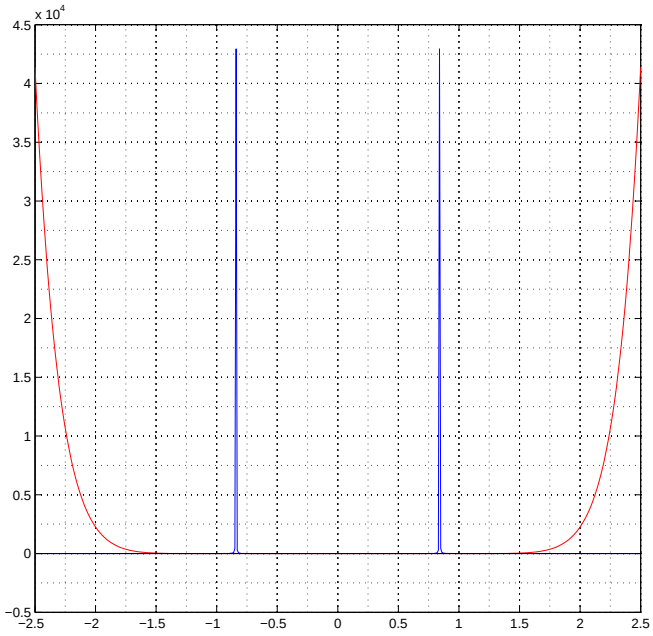


FIG. 12: Plot of psolq(r) , $f(\lambda) = p(\lambda)p(\lambda)^* - 1$; (blue) $aq^2 + bq + c = 0$, and, corresponding characteristic 6th degree polynomial function, $f(\lambda) = \lambda^{2\cdot 6} + \alpha_5\lambda^{2\cdot 5} + \alpha_4\lambda^{2\cdot 4} + \alpha_3\lambda^{2\cdot 3} + \alpha_2\lambda^{2\cdot 2} + \alpha_1\lambda^{2\cdot 1} + \alpha_0$, (red), showing the contrast in the general plots, when scanning the range visually to find and narrow down the location of the roots, for the quaternion quadratic with parameters, $a = 2 + i - k$, $b = -3 + j + k$, $c = 1 - j + k$.

.

```
>> rr = -2.5:.01:2.5;
>> size(rr)
ans =
1 501
>> zz = polyval(vv,rr);
>> for i = 1:501
yy(i)=psolq(rr(i));
end
>> plot(rr,yy,'b',rr,zz,'r')
>> hold on
>> plot([-2.5 2.5],[0 0],'k-')
>> grid on
>> grid minor
```

Of the "six" roots, provided by the `roots(vv)` function, only two are real. The other four are complex numbers. These extra roots are the "*bi-quaternion*" solutions. We can't handle bi-quaternion calculations with our 2×2 complex matrix representation, because we're already using complex variables for another purpose, to encode the quaternion. So, to see the bi-quaternion root solutions, we need to write another routine that doesn't use complex representation, so that complex numbers can be used instead for the bi-quaternion coefficients. Here's the modified biquaternion enabled code for the provisional unit direction function.

in MATLAB code:

```
function P=psolqvc(A,B,C,r)
% Bi-Quaternion Unit Direction Function ver:  1.0, (aqq + bq + c = 0)
%
%
%    r.(r^2.a* - c*).(cb*.(r^2.a - c) - r^2.(r^2.a - c).a*b)
% p = ------------------------------------------------------------
%           (r^4.|a|^2 - |c|^2).|r^2.a - c|^2
%
%
  NP1 = r*(r^2*conq(A) - conq(C));
  NP2 = mulq(C,mulq(conq(B),(r^2*A - C))) - r^2*mulq((r^2*A - C),mulq(conq(A),B));
  NP = mulq(NP1,NP2);
  DP1 = (r^4*mulq(A,conq(A)) - mulq(C,conq(C)));
  DP2 = mulq(r^2*A - C, conq(r^2*A - C));
  DP = mulq(DP1,DP2);
  P = NP/DP(1);
end
```

The function, `P=psolqvc(A,B,C,r)`, takes in the quadratic equation parameters in 'row-vector' format, `(A,B,C)`, and the scalar magnitude, which is now a complex number, $\lambda = r \in \mathbb{C}$, and returns a row-vector 'v', with complex 'c' coefficients this time (hence the '`vc`' ending), instead of real coefficients returned by the previous, `psolqv(r)`, routine. This allows us to carefully conjugate just the quaternion parts, with our function, `conq(Q)`, while not conjugating the complex parts, so we get the right result. Other than that, it's the same formula as before; Using one of our optimized versions for simplicity. Then the six solutions, including the bi-quaternions this time, are;

```
>> rt
rt =
1.2013 + 0.0000i
0.4972 + 0.6782i
0.4972 - 0.6782i
0.7444 + 0.3910i
0.7444 - 0.3910i
0.5886 + 0.0000i
>>
>> p1 = psolqvc(A,B,C,rt(1))                p1 = 0.8985 -0.0844 -0.3973 -0.1666
>> p2 = psolqvc(A,B,C,rt(2))                p2 = 0.9422 - 0.1694i -0.0905 + 0.7192i 0.2226 + 0.1000i 0.8195 + 0.2470i
>> p3 = psolqvc(A,B,C,rt(3))                p3 = 0.9422 + 0.1694i -0.0905 - 0.7192i 0.2226 - 0.1000i 0.8195 - 0.2470i
>> p4 = psolqvc(A,B,C,rt(4))                p4 = 0.6293 - 0.2938i 0.2191 - 0.6706i -1.0941 - 0.2325i 0.2366 + 0.3269i
>> p5 = psolqvc(A,B,C,rt(5))                p5 = 0.6293 + 0.2938i 0.2191 + 0.6706i -1.0941 + 0.2325i 0.2366 - 0.3269i
>> p6 = psolqvc(A,B,C,rt(6))                p6 = 0.1484 0.2541 -0.5570 0.7766
>>
>> q1 = rt(1)*p1                q1 = 1.0793 -0.1014 -0.4773 -0.2001
>> q2 = rt(2)*p2                q2 = 0.5833 + 0.5548i -0.5327 + 0.2962i 0.0429 + 0.2006i 0.2399 + 0.6786i
>> q3 = rt(3)*p3                q3 = 0.5833 - 0.5548i -0.5327 - 0.2962i 0.0429 - 0.2006i 0.2399 - 0.6786i
>> q4 = rt(4)*p4                q4 = 0.5833 + 0.0273i 0.4253 - 0.4136i -0.7236 - 0.6010i 0.0483 + 0.3359i
>> q5 = rt(5)*p5                q5 = 0.5833 - 0.0273i 0.4253 + 0.4136i -0.7236 + 0.6010i 0.0483 - 0.3359i
>> q6 = rt(6)*p6                q6 = 0.0873 0.1496 -0.3279 0.4571
```

We can check that our provisional unit direction quaternions, p, are really unit numbers, using, `p*p.'`, where we have to apply the "dot version" of the operator $'$ to tell MATLAB to 'transpose', but not 'conjugate' the vector, in evaluating the expression. Then we can confirm the unit;

```
>> p1*p1.'                                   ans = 1.0000
>> p2*p2.'                                   ans = 1.0000 - 0.0000i
>> p3*p3.'                                   ans = 1.0000 + 0.0000i
>> p4*p4.'                                   ans = 1.0000 + 0.0000i
>> p5*p5.'                                   ans = 1.0000 - 0.0000i
>> p6*p6.'                                   ans = 1.0000
>>
```

Each of the six roots satisfies the constraint, $|p| = 1$, and after building our solutions, `q = r*p`, we can do the usual, and `Verify results...`

```
>> v1 = mulq(A,mulq(q1,q1)) + mulq(B,q1) + C
>> v2 = mulq(A,mulq(q2,q2)) + mulq(B,q2) + C
>> v3 = mulq(A,mulq(q3,q3)) + mulq(B,q3) + C
>> v4 = mulq(A,mulq(q4,q4)) + mulq(B,q4) + C
>> v5 = mulq(A,mulq(q5,q5)) + mulq(B,q5) + C
>> v6 = mulq(A,mulq(q6,q6)) + mulq(B,q6) + C
>>
v1 = 1.0e-14 * 0.2887 -0.0222 -0.3331 0.2887
v2 = 1.0e-14 * -0.1332 + 0.0888i -0.1110 + 0.1055i 0.1776 + 0.0222i 0.0000 + 0.0444i
v3 = 1.0e-14 * -0.1332 - 0.0888i -0.1110 - 0.1055i 0.1776 - 0.0222i 0.0000 - 0.0444i
v4 = 1.0e-14 * 0.1554 + 0.0666i -0.0111 + 0.0444i -0.1776 + 0.0222i 0.1554 - 0.0222i
v5 = 1.0e-14 * 0.1554 - 0.0666i -0.0111 - 0.0444i -0.1776 - 0.0222i 0.1554 + 0.0222i
v6 = 1.0e-15 * -0.6661 0 0.4441 -0.4441
>>
```

All solutions evaluate the quadratic expression, $aq^2 + bq + c$, to 0, at the level of, 10^{-15}, confirming the solutions. Hence we have our quadratic equation, and it's complete solution with all six bi-quaternion roots, given below.

BIQUATERNION SOLUTIONS:

$$(2 + i - k)q^2 + (-3 + j + k)q + (1 - j + k) = 0 \tag{1.713}$$

$$\begin{aligned}
q_1 &= (1.0793)\cdot 1 + (-0.1014)\cdot \boldsymbol{i} + (-0.4773)\cdot \boldsymbol{j} + (-0.2001)\cdot \boldsymbol{k}; & \lambda_1 &= 1.2013 + 0.0000\iota \\
q_2 &= (0.5833 + 0.5548\iota)\cdot 1 + (-0.5327 + 0.2962\iota)\cdot \boldsymbol{i} + (0.0429 + 0.2006\iota)\cdot \boldsymbol{j} + (0.2399 + 0.6786\iota)\cdot \boldsymbol{k}; & \lambda_2 &= 0.4972 + 0.6782\iota \\
q_3 &= (0.5833 - 0.5548\iota)\cdot 1 + (-0.5327 - 0.2962\iota)\cdot \boldsymbol{i} + (0.0429 - 0.2006\iota)\cdot \boldsymbol{j} + (0.2399 - 0.6786\iota)\cdot \boldsymbol{k}; & \lambda_3 &= 0.4972 - 0.6782\iota \\
q_4 &= (0.5833 + 0.0273\iota)\cdot 1 + (0.4253 - 0.4136\iota)\cdot \boldsymbol{i} + (-0.7236 - 0.6010\iota)\cdot \boldsymbol{j} + (0.0483 + 0.3359\iota)\cdot \boldsymbol{k}; & \lambda_4 &= 0.7444 + 0.3910\iota \\
q_5 &= (0.5833 - 0.0273\iota)\cdot 1 + (0.4253 + 0.4136\iota)\cdot \boldsymbol{i} + (-0.7236 + 0.6010\iota)\cdot \boldsymbol{j} + (0.0483 - 0.3359\iota)\cdot \boldsymbol{k}; & \lambda_5 &= 0.7444 - 0.3910\iota \\
q_6 &= (0.0873)\cdot 1 + (0.1496)\cdot \boldsymbol{i} + (-0.3279)\cdot \boldsymbol{j} + (0.4571)\cdot \boldsymbol{k}; & \lambda_6 &= 0.5886 + 0.0000\iota
\end{aligned} \tag{1.714}$$

Since the quaternion magnitudes are now complex numbers, these magnitudes themselves have their own magnitudes, which are complex number amplitudes of the various $\lambda's$. So, let's take a look at these second order magnitudes:

```
>> sqrt(rt(1)*rt(1)')                        ans = 1.2013
>> sqrt(rt(2)*rt(2)')                        ans = 0.8409
>> sqrt(rt(3)*rt(3)')                        ans = 0.8409
>> sqrt(rt(4)*rt(4)')                        ans = 0.8409
>> sqrt(rt(5)*rt(5)')                        ans = 0.8409
>> sqrt(rt(6)*rt(6)')                        ans = 0.5886
>>
```

We see that the four truely bi-quaternion solutions, have magnitudes with amplitudes equal to the geometric mean of the real quaternion root magnitudes! That is, $|\lambda_2| = |\lambda_3| = |\lambda_4| = |\lambda_5| = \sqrt{\lambda_1 \cdot \lambda_6} = \sqrt{|c|/|a|}$, which is again that critical point, $\lambda_0 = \sqrt{|c|/|a|}$ `= 0.8049`, where the denominator vanishes.

Now you might think, that since, this "6th" degree polynomial, is really an expression in the *square magnitude*, λ^2, that when we take the square root, to ultimately find λ, we should have two roots from the $\pm$ signs, thus really should be recording 12 roots for these magnitudes, and hence possibly 12 quaternion solutions! However, notice that the provisional unit direction function is an odd function of λ, i.e., $p(-\lambda) = -p(\lambda)$, so that while we could have 12 magnitudes, considering the 6 pairs of $\pm\lambda$ results, each opposite pair produces the very same quaternion

solution, $q = (+\lambda) \cdot p(+\lambda) = (-\lambda) \cdot p(-\lambda)$, so that there are still only six unique roots. That leaves us with a bit of an ambiguous situation. Technically, we now allow magnitudes to be negative or positive, because they are complex number values, so we could speak of 12 roots. But then we'd always have to pair them up, to recognise that only six are distinct. Hence, it's a matter of covention. Hamilton chose to identify six roots. We shall do the same. So then, how do we pick which of the pair of magnitudes will be our "positive" magnitude? We follow MATLAB's root solver and pick the value with positive real component, $Re[\lambda] > 0$.

`Left-Hand Solutions.` But, there's nothing that requires the iota symbol, $\iota = \sqrt{-1}$, to be particularly "the" complex number unit. It only has to obey the same rules, commutating with the right hand elements, $\{i, j, k\}$. We could just as well define, $\iota = n_1 i' + n_2 j' + n_3 k'$, with, $i', j', k' \in \mathbb{H}_L,\ n_1, n_2, n_3 \in \mathbb{R},\ n_1^2 + n_2^2 + n_3^2 = 1$, and use our left hand numbers instead. In which case, of course, we'd have an infinite number of "these" biquaternion solutions. We'll leave the proper discussion of biquaternion and alternative solutions for a future paper, there being enough on the plate at the moment just looking into the details of the ordinary quaternions over the field of real numbers.

Is there another way to find the λ magnitudes?

$$\begin{aligned}
\text{Form \#1:}\quad & aq^2 + bq + c = 0 && (1.715)\\
& \uparrow \\
& aq^2 = -(bq + c) && (1.716)\\
& |aq^2|^2 = |bq + c|^2 && (1.717)\\
& (aq^2)(aq^2)^* = (bq + c)(bq + c)^* && (1.718)\\
\text{Let,}\quad & q = \lambda p\ , \quad \text{where, } \lambda \in \mathbb{R},\ \lambda > 0,\ p \in \mathbb{H},\ |p| = 1 && (1.719)\\
\therefore\quad & \lambda^4|a|^2 = \lambda^2|b|^2 + |c|^2 + \lambda bpc^* + \lambda cp^*b^* && (1.720)\\
& cp^*b^* = -bpc^* + (\lambda^4|a|^2 - \lambda^2|b|^2 - |c|^2)/\lambda && (1.721)\\
& p^* = -(1/c)bpc^*(1/b^*) + (1/c)(1/b^*)(\lambda^4|a|^2 - \lambda^2|b|^2 - |c|^2)/\lambda && (1.722)\\
\text{But,}\quad & pp^* = p^*p = 1 && (1.723)\\
\therefore\quad & -(1/c)bpc^*(b/|b|^2)p + (1/c)(b/|b|^2)(\lambda^4|a|^2 - \lambda^2|b|^2 - |c|^2)\lambda^{-1}p = 1 && (1.724)\\
& pc^*bp - (\lambda^4|a|^2 - \lambda^2|b|^2 - |c|^2)\lambda^{-1}p + b^*c = 0 && (1.725)\\
& \downarrow \\
\text{Form \#3:}\quad & qaq + bq + c = 0 && (1.726)
\end{aligned}$$

The quadratic equation form, $aq^2 + bq + c = 0$, can be transformed into the alternate form, $qaq + bq + c = 0$, by introducing the "squaring" operation early in the manipulations; in this case, it's the absolute square. This alternate form shall be discussed more fully later in a separate section below. But, because the 'b' in the new form #3 equation is just a scalar, in this case, we can solve this readily right here. To do this, let us define, $\Lambda = (\lambda^4|a|^2 - \lambda^2|b|^2 - |c|^2)/(2\lambda)$, noting, $\Lambda \in \mathbb{R}$. Then the new quadratic becomes,

$$pc^*bp - 2\Lambda p + b^*c = 0\ , \qquad \Lambda \in \mathbb{R} \tag{1.727}$$

mul on the left by, c^*b,

$$\begin{aligned}
& c^*bpc^*bp - 2\Lambda c^*bp + c^*bb^*c = 0 && (1.728)\\
& (c^*bp)^2 - 2\Lambda c^*bp + |b|^2|c|^2 = 0 && (1.729)\\
& (c^*bp - \Lambda)^2 - \Lambda^2 + |b|^2|c|^2 = 0 && (1.730)
\end{aligned}$$

$$c^*bp = \Lambda \pm \sqrt{\Lambda^2 - |b|^2|c|^2} \tag{1.731}$$

$$\therefore \quad p = \frac{b^*c}{|b|^2|c|^2}\left(\Lambda \pm \sqrt{\Lambda^2 - |b|^2|c|^2}\right) \tag{1.732}$$

Now, there are three possibilities for the discriminant under the root sign, leading to three cases.

$$\text{case 1: } \Lambda^2 - |b|^2|c|^2 > 0, \qquad p = \frac{b^*c}{|b|^2|c|^2}\left(\Lambda \pm 1 \cdot \sqrt{\Lambda^2 - |b|^2|c|^2}\right) , \tag{1.733}$$

$$\text{case 2: } \Lambda^2 - |b|^2|c|^2 = 0, \qquad p = \frac{b^*c}{|b|^2|c|^2}\Lambda \;=\; \pm\frac{b^*c}{|b||c|} , \tag{1.734}$$

$$\text{case 3: } \Lambda^2 - |b|^2|c|^2 < 0, \qquad p = \frac{b^*c}{|b|^2|c|^2}\left(\Lambda + u \cdot \sqrt{|b|^2|c|^2 - \Lambda^2}\right) , u = n_1 i + n_2 j + n_3 k \, , |u| = 1. \tag{1.735}$$

The first thing we need to check, is the conditions under which these candidate solutions actually result in the unit quaternion, so, $pp^* = 1$. Inspecting case 1, first, we find,

$$pp^* = \frac{b^*cc^*b}{|b|^4|c|^4}\left(\Lambda \pm \sqrt{\Lambda^2 - |b|^2|c|^2}\right)^2 \tag{1.736}$$

$$= \frac{1}{|b|^2|c|^2}\left(\Lambda^2 \pm 2\Lambda\sqrt{\Lambda^2 - |b|^2|c|^2} + \Lambda^2 - |b|^2|c|^2\right) \tag{1.737}$$

$$\text{so that, } pp^* = 1, \quad \Longrightarrow \quad \mp\Lambda\sqrt{\Lambda^2 - |b|^2|c|^2} = \Lambda^2 - |b|^2|c|^2 \tag{1.738}$$

Squaring both sides we find,

$$\Lambda^2(\Lambda^2 - |b|^2|c|^2) \;= \left(\Lambda^2 - |b|^2|c|^2\right)^2 \tag{1.739}$$

$$\therefore \;\; \Lambda^2 = \Lambda^2 - |b|^2|c|^2 \tag{1.740}$$

This last equation is only true when, $b = 0$, or, $c = 0$, or, $b = c = 0$, in which case, the quadratic equation, $aq^2 + bq + c = 0$, reduces to, $aq^2 + c = 0$, or, $(aq - b)q = 0$, or, $aq^2 = 0$, none of which have solutions exploiting the candidate form, p, shown here. So, we can toss out case 1, since it gives no valid solutions. Next we look at case 2,

$$pp^* = \left(\pm\frac{b^*c}{|b||c|}\right)\left(\pm\frac{c^*b}{|b||c|}\right) \;=\; \frac{|b|^2|c|^2}{|b|^2|c|^2} \;=\; 1 \tag{1.741}$$

This suggests that we could possibly have a solution of the form,

$$q \;=\; \lambda p \;=\; \lambda\left(\pm\frac{b^*c}{|b||c|}\right) \tag{1.742}$$

But, would this expression be consistent with the quadratic equation, $aq^2 + bq + c = 0$? Let's see,

$$\lambda^2 a\frac{b^*cb^*c}{|b|^2|c|^2} \pm \lambda b\frac{b^*c}{|b||c|} + c = 0 \tag{1.743}$$

$$\lambda^2 ab^*cb^*c \pm \lambda|b|^3|c|c + |b|^2|c|^2c = 0 \tag{1.744}$$

$$(\lambda^2 ab^*cb^* \pm \lambda|b|^3|c| + |b|^2|c|^2)c = 0 \tag{1.745}$$

$$\therefore \;\; ab^*cb^* \;\in\; \mathbb{R}, \;\; \text{so, } \; ab^*cb^* = \pm|a||b|^2|c|$$

so we have,

$$\pm\lambda^2|a||b|^2|c| \pm \lambda|b|^3|c| + |b|^2|c|^2 = 0 \tag{1.746}$$

i.e.,

$$\pm\lambda^2|a| \pm \lambda|b| + |c| = 0 \tag{1.747}$$

As we see from the results above, when we factor out the quaternion, c, from the quadratic form, we're left with an expression that has the sum of two scalar terms plus a quaternion proportional to, ab^*cb^*. So if, $c \neq 0$, all these terms within the parenthesis must be scalar, if they are to offset each other to produce a total sum of 0. Hence, ab^*cb^*, must be a scalar, which in our current context is just some real valued number, $ab^*cb^* \in \mathbb{R}$. Now we presume this product is non-zero, since we're assuming, $b \neq 0, c \neq 0$, here. If, $a = 0$, then the quadratic equation reduces to the linear equation, $bq + c = 0$, which has solution, $q = -c/b$, with, $q = \lambda p$, then yielding, $\lambda = |c|/|b|$, which can also be determined from the above factor. However, the interesting case is when, $a \neq 0, i.e.\; ab^*cb^* \in \mathbb{R}\backslash\{0\}$.

The magnitude of the real valued scalar factor, ab^*cb^*, is readily determined, but the sign is unknown, so we write, $ab^*cb^* = \pm|a||b|^2|c|$. Of course, for a given set of known parameters $\{a, b, c\}$, the sign is completely determined, but, at the moment, we must consider that we don't know what that sign is while investigating the possibilities. The first $\pm$ sign pair, appearing in the square term of the quadratic, is not necessarily correlated with the second $\pm$ sign pair, prefixing the linear term, in some definite way. The first sign is immediately known, given a particular quadratic to solve, but the second one is yet to be determined, and depends on the value of λ found to satisfy the equation. Hence, this last scalar quadratic equation in λ, should really be broken out into four distinct equations for clarity.

$$+|a|\lambda^2 + |b|\lambda + |c| = 0 \ , \qquad ab^*cb^* > 0, \ \ \Lambda = +|b||c| \tag{1.748}$$
$$+|a|\lambda^2 - |b|\lambda + |c| = 0 \ , \qquad ab^*cb^* > 0, \ \ \Lambda = -|b||c| \tag{1.749}$$
$$-|a|\lambda^2 + |b|\lambda + |c| = 0 \ , \qquad ab^*cb^* < 0, \ \ \Lambda = +|b||c| \tag{1.750}$$
$$-|a|\lambda^2 - |b|\lambda + |c| = 0 \ , \qquad ab^*cb^* < 0, \ \ \Lambda = -|b||c| \tag{1.751}$$

$$\Lambda \ = \ (\lambda^4|a|^2 - \lambda^2|b|^2 - |c|^2)/(2\lambda) \ = \ \pm|b||c| \tag{1.752}$$

Each of the four *quadratic* equations here, is paired with a particular corresponding *quartic* equation, from the discriminant constraint, $\Lambda = \pm|b||c|$, forming a pair of simultaneous polynomial equations, one of order 2, and the other of order 4, that must both be satisfied by any λ that puports to solve the original quadratic equation, $aq^2 + bq + c = 0$. The quartic equation can be re-written,

$$|a|^2\lambda^4 - |b|^2\lambda^2 - 2|b||c|\lambda - |c|^2 \ = \ -(+|a|\lambda^2 + |b|\lambda + |c|)(-|a|\lambda^2 + |b|\lambda + |c|) \ = \ 0 \ , \quad \Lambda = +|b||c| \tag{1.753}$$
$$|a|^2\lambda^4 - |b|^2\lambda^2 + 2|b||c|\lambda - |c|^2 \ = \ -(+|a|\lambda^2 - |b|\lambda + |c|)(-|a|\lambda^2 - |b|\lambda + |c|) \ = \ 0 \ , \quad \Lambda = -|b||c| \tag{1.754}$$

We see that in each pair of polynomial equations, the quadratic form is contained in the quartic form, and since the candidate, λ, must simultaneously be a solution of both polynomial equations, there are only two such values, already given by the quadratic, and the quartic equation can be dispensed with, as the two additional solutions it suggests don't fit the corresponding quadratic for the particular conditions. This means we only have to consider the quadratic equations, and examine the pair of quadratics that correspond to the appropriate sign in, $ab^*cb^* = \pm|a||b|^2|c|$.

When, $ab^*cb^* \in \mathbb{R}^+\backslash\{0\}$,

$$\lambda = \left| \frac{+|b| \pm \sqrt{|b|^2 - 4|a||c|}}{2|a|} \right| \ , \qquad ab^*cb^* = +|a||b|^2|c| > 0 \ , \ |b|^2 \geq 4|a||c| \tag{1.755}$$
$$q = \lambda p = \left| \frac{+|b| \pm \sqrt{|b|^2 - 4|a||c|}}{2|a|} \right| \cdot \left(+ \frac{b^*c}{|b||c|} \right) \tag{1.756}$$

The problem with this strategy, however, is that every time we *square* a parameter we lose information. We might know that, $q = -1$, but when we compute, $q^2 = (-1)^2 = +1$, we lose this knowledge. Now, when we try to get back the original fact, we take the square-root, $q = \sqrt{q^2} = \sqrt{+1} = \pm 1$. We get two possibilities. We find multiple candidates for the parameter. The solution is now ambiguous. We were certain before taking the square, but after the square we no longer know some of the specific detail. This is particularly problematic in quaternions, where we can go from "one," $q = i$, through the "square," $q^2 = (i)^2 = -1$, back down through the "square-root" to an "infinite" number of possibilities, $q = \sqrt{q^2} = \sqrt{-1} = n$, where, $n = (n_1 i + n_2 j + n_3 k) \in \mathbb{H},\ n^2 = -(n_1^2 + n_2^2 + n_3^2) = -1, n_1, n_2, n_3 \in \mathbb{R}$. It should come as no surprise then, that when we transform our quadratic equation above, the new form suggests many candidate solutions that are not really solutions to the original equation! Here, we didn't just take a "square," we took the "modulus squared," i.e. $|aq^2|^2 = |bq + c|^2$, which, nevertheless, has the same effect of supressing information, and introducing spurious multiplicity.

We deal with the particular form, $qaq + bq + c = 0$, later in section #3 below. However, in this particular situation we notice that the linear coefficient, b, in this transformed equation, is just a scalar, $b \to -(\lambda^4|a|^2 - \lambda^2|b|^2 - |c|^2)/\lambda \in \mathbb{R}$, which enables us to readily solve the equation right here, by using the usual *completing the square* method.

Solving:

$$c^*bpc^*bp - (\lambda^4|a|^2 - \lambda^2|b|^2 - |c|^2)\lambda^{-1}c^*bp + c^*bb^*c = 0 \tag{1.757}$$

$$\left(c^*bp - \frac{\lambda^4|a|^2 - \lambda^2|b|^2 - |c|^2}{2\lambda}\right)^2 - \left(\frac{\lambda^4|a|^2 - \lambda^2|b|^2 - |c|^2}{2\lambda}\right)^2 + |b|^2|c|^2 = 0 \tag{1.758}$$

$$c^*bp = \left(\frac{\lambda^4|a|^2 - \lambda^2|b|^2 - |c|^2}{2\lambda}\right) \pm \sqrt{\left(\frac{\lambda^4|a|^2 - \lambda^2|b|^2 - |c|^2}{2\lambda}\right)^2 - |b|^2|c|^2} \tag{1.759}$$

$$p = \frac{b^*c}{|b|^2|c|^2} \cdot \left[\left(\frac{\lambda^4|a|^2 - \lambda^2|b|^2 - |c|^2}{2\lambda}\right) \pm u \cdot \sqrt{\left|\left(\frac{\lambda^4|a|^2 - \lambda^2|b|^2 - |c|^2}{2\lambda}\right)^2 - |b|^2|c|^2\right|}\right] \tag{1.760}$$

$$\text{Where,} \quad u = \begin{matrix} +1, & D(\lambda) \geq 0 \\ n, & D(\lambda) < 0 \end{matrix}, \qquad \text{with,} \quad D(\lambda) = \left(\frac{\lambda^4|a|^2 - \lambda^2|b|^2 - |c|^2}{2\lambda}\right)^2 - |b|^2|c|^2 \tag{1.761}$$
$$n = (n_1 i + n_2 j + n_3 k) \in \mathbb{H},\ n^2 = -1.$$

Depending on whether the discriminant, $D(\lambda)$, is positive or negative, the unit, u, is real or imaginary. When, u, is imaginary, it has an infinite number of possibilities. So, often, we may start out with an equation, $aq^2 + bq + c = 0$, which has just 2 solutions, but after the transformation, the new form, $qaq + bq + c = 0$, presents an infinity of candidate solutions within which those two original solutions are hidden, and we are unable to identify and extract them to solve the original equation. The norm of the factor in square brackets is just, $|[\,]| = |b||c|$, so that, $|p| = (|b^*c|/(|b|^2|c|^2)) \cdot |b||c| = 1$, and so, setting $|p(\lambda)| = 1$, here, doesn't give us any special information on λ, that would enable us to pick particular values out from the infinity of possibilities. By *squaring* we lose all the information contained in the original quadratic equation. This suggests that, when transforming quaternion expressions, we must take care in applying only *linear transformations* on the unknown variables, q, even though we may square the known parameters, like a, b, c, else we risk losing all the necessary information required to find the particular solutions.

Our *two-hand two-step* method works, largely because it consists mostly of *linear transformation manipulations* on the unknown variable, q. The initial step, multiplying by the inverse, $1/q = q^*$, effectively "*linearizes*" the quadratic form, $aq^2 + bq + c$, to create a corresponding linear form, $aq + b + cq^*$, in the two variables, q and q^*. Thereafter, all the operations that transform the expressions remain linear in these two unknowns. The strategy works, because it avoids the problematic situation of the non-reversable "squaring" transformation, which generally destroys essential information in quaternion algebraic manipulations, as just illustrated above. This highlights one of the key difficulties in quaternion algebra, in that manipulations of algebraic expressions are much more tricky, and challenging to accomplish successfully, than in real or complex variable algebra, where things are more sedate.

Of course, there are situations in mathematics where squaring, and other operations that surpress detailed information, can be very useful. Such cases are found in statistics, neural networks, and machine learning, where computing things like the mean and variance, or other aggregate variables, from a detailed data set, will also hide particular information, at the benefit of simplification by parameterization. But, in solving polynomial equations we are generally trying to do the exact opposite, and find the detailed information for the problem at hand. The quadratic equation form itself, $aq^2 + bq + c = 0$, is a kind of "parameterized expression" already, and we're attempting to find the specific detailed data points that fit it. So, any operations that further supress this kind of information will compound the difficulty in finding the actual solutions, by introducing other unlrelated data points into the mix. In real and complex variable algebra, this squaring isn't so much of a problem, since, at most, the multiplicity introduced is limited to just two choices, $\pm$, one of which, at least, must be our solution. But, in quaternions, we can get an infinite set to choose from, which is the real problematic issue dramatically changing conditions.

The key thing, then, in quaternion algebra, is to keep to *linear transformations* on the unknown variables, as much as possible, in expression manipulations.

COMBINING CASES:

In the refernce section below, we give the complete solution, with all detail presented in one place. However, there's still something else that we can do to simplify things. We separated the quaternion quadratic equation into two systems, one with, $b = 0$, and the other, $b \neq 0$. These two are tackeled differently, with separate and distinct methodologies. However, we notice that they both have similar 4-part case solutions, and the reader might have observed that it's possible to combine these two disjoint sets into just one set, by merging the resultant solutions of the quadratic equation. So, although two distinct methods are used, we can make the final result appear to have just come out of a single coherent methodology. This is our final optimization.

$b = 0:$

$$aq^2 + c = 0 \tag{1.762}$$

$$\texttt{Case 1: } a^*c \in \mathbb{R},\ a^*c > 0.: \qquad q = \sqrt{|c|/|a|} \cdot (n_1 i + n_2 j + n_3 k),\ \texttt{``out of scope''} \infty \texttt{\# roots!} \tag{1.763}$$

$$\texttt{Case 2: } a^*c \in \mathbb{R},\ a^*c < 0.: \qquad q = \pm\sqrt{|c|/|a|} \cdot 1 \tag{1.764}$$

$$\texttt{Case 3: } a^*c \notin \mathbb{R},\ S(a^*c) = 0.: \quad q = \pm\sqrt{|c|/|a|} \cdot (1 - a^*c/|a^*c|)/\sqrt{2} \tag{1.765}$$

$$\texttt{Case 4: } a^*c \notin \mathbb{R},\ S(a^*c) \neq 0.: \quad q = \pm\sqrt{|c|/|a|} \cdot (1 - a^*c/|a^*c|/)/\sqrt{2 - (a^*c + (a^*c)^*)/|a^*c|} \tag{1.766}$$

$b \neq 0:$

$$aq^2 + bq + c = 0 \tag{1.767}$$

$$w = \lambda^2 a^*(a(\lambda p)^2 + b(\lambda p) + c)\ , \qquad \lambda = \sqrt{|c|/|a|} \tag{1.768}$$

$$\text{where, } u = u(w): \tag{1.769}$$

$$\texttt{Case 1: } w \in \mathbb{R},\ w > 0.: \qquad u = (n_1 i + n_2 j + n_3 k),\ \texttt{``within scope''} \infty \texttt{\# roots!} \tag{1.770}$$

$$\texttt{Case 2: } w \in \mathbb{R},\ w < 0.: \qquad u = 1 \tag{1.771}$$

$$\texttt{Case 3: } w \notin \mathbb{R},\ S(w) = 0.: \quad u = (1 - w/|w|)/\sqrt{2} \tag{1.772}$$

$$\texttt{Case 4: } w \notin \mathbb{R},\ S(w) \neq 0.: \quad u = (1 - w/|w|/)/\sqrt{2 - (w + w^*)/|w|} \tag{1.773}$$

Notice that the prefactor, λ^2, in the definition of the weighted quadratic residual, w, can be removed without affecting the solution. Thus, we may re-define, $w = a^*(a(\lambda p)^2 + b(\lambda p) + c$. So, now when, $p = 0$, we have, $w = a^*c$. Then, all the "within scope" w-cases, become exactly the same as the "out of scope" a^*c-cases. All of the "*same magnitudes*" solutions for both systems, $b = 0$ and $b \neq 0$, are then given by the `offplane` magic formula;

$$q = \lambda \cdot \left(p \pm u \cdot \sqrt{\frac{|a(\lambda p)^2 + b(\lambda p) + c|}{|c|}} \right) \tag{1.774}$$

Thus, we only have to "simplify" the weight, w, and remember to set, $p = 0$, whenever, $b = 0$, to achive this optimization. We can easily accept the extra $\pm$ specification on the infinite roots case, when, $b = 0$. We can also now combine "`Case 3`" and "`Case 4`," so the original 8-part case set becomes just a 3-part case solution. *But, whenever we optimize, we also hide some details!* We implement this "merge combination" in our final code of this section #1: The, MATLAB code `[Q1 Q2]=aqqbqc(A,B,C)`, version 1.1, for the `(aqq + bq + c = 0)` quadratic form. This final function is also written to be independent of all the other code functions we've generally used in exploring solutions in this paper. It only calls the standard `MATLAB` functions. So, it can be used as a convenient standalone tool.

Fixup Factor:

Are further optimizations and/or simplifications possible? Let's take a look at the "fixup" factors used in the solutions. Writing, F, for the fixup, which is always a scalar, i.e., $F \in \mathbb{R}$, we have;

$$q = \lambda \cdot (p \pm u \cdot F) \tag{1.775}$$

$$(p + u \cdot F)(p + u \cdot F)^* = 1 \tag{1.776}$$
$$F^2 + F \cdot (up^* + pu^*) + |p|^2 - 1 = 0 \tag{1.777}$$

$$(p - u \cdot F)(p - u \cdot F)^* = 1 \tag{1.778}$$
$$F^2 - F \cdot (up^* + pu^*) + |p|^2 - 1 = 0 \tag{1.779}$$

$$\therefore \quad 2F^2 + 2(|p|^2 - 1) = 0 \tag{1.780}$$
$$F^2 = 1 - |p|^2 = \frac{|a(\lambda p)^2 + b(\lambda p) + c|}{|c|} \tag{1.781}$$

$$q = \lambda \cdot \left(p \pm u \cdot \sqrt{\frac{|a(\lambda p)^2 + b(\lambda p) + c|}{|c|}} \right) \quad = \quad \lambda \cdot \left(p \pm u \cdot \sqrt{1 - |p|^2} \right) \tag{1.782}$$
$$= \lambda \cdot \left(p \pm u \cdot |p| \sqrt{|p|^{-2} - 1} \right) \tag{1.783}$$
$$= \lambda \cdot p \cdot \left(1 \pm p^{-1} u \cdot |p| \sqrt{|p|^{-2} - 1} \right) \tag{1.784}$$
$$= \lambda \cdot p \cdot \left(1 \pm \frac{p^* u}{|p|^2} \cdot |p| \sqrt{|p|^{-2} - 1} \right) \tag{1.785}$$

We make use of the fact that, $(p \pm u \cdot F)$, is always a unit quaternion, to find the relations between the various parameters. This allows for a simplification of formulas. However, we don't gain any advantage in computation efficiency, since we still have to compute that weighted quadradic residual, $w = a(\lambda p)^2 + b(\lambda p) + c$, even this simplified form, which is used to decide which of the four cases in the 4-part case set we need for the unit, u. So, since we have to calculate this residual anyway, we might as well use it to construct the fixup factor, $F = \sqrt{|w|/|c|}$. We even get a little bit of optimization, by recognizing that since, $\lambda = \sqrt{|c|}$, in the "lead reduced" variables, we can then have, $\lambda \cdot u \sqrt{|w|/|c|} = u\sqrt{|w|}$, saving a few calculation steps. But, by re-arranging the fixup expressions we do get to compare the `onplane` and `offplane` magic factor solutions a bit better, putting them on a more equal footing.

Magic Factors:

$$\texttt{offplane:} \quad q = \lambda \cdot p \cdot \left(1 \quad \mp \quad \frac{p^* u}{|p|} \cdot \sqrt{|p|^{-2} - 1} \right) \tag{1.786}$$

$$\texttt{onplane:} \quad q = \lambda \cdot p \cdot \left(1 \quad \pm \quad \frac{p - p^*}{|p - p^*|} \cdot \sqrt{|p|^{-2} - 1} \right) \qquad \rightarrow \quad u = -\frac{pp - pp^*}{|pp - pp^*|} \tag{1.787}$$

[Note the flipped signs, $\mp$ vs. $\pm$, required to match up the expressions!]

Here we see that the `onplane` magic factor effectively employs the unit, $u = -(pp - pp^*)/|pp - pp^*|$, to do the same job in the `offplane` result. This bit of additional knowledge can then be used to modify the code implementation, of `[Q1 Q2]=aqqbqc(A,B,C)`, to enable the user to "force an onplane" solution in situations where numerical imprecision on input parameters imperfectly describe the problem under consideration. This would require adding another parameter to the input, e.g. `[Q1 Q2]=aqqbqc(A,B,C,onplane)`, to select this option in code. We leave this up to the reader to implement, if so desired. Our code functions just return the best numerical solutions, based on smallest residual values, which may not be what the user wants for his particular study.

`Magic Factors Above the Axis?`

In our theory, developed in the many pages above, we've seen that when the function, $f(\lambda) = p(\lambda)p(\lambda)^* - 1$, has the form of a "*hill top*" below the horizontal axis, we may *then* apply the `magic factor` solution method, which doesn't require any numerical search for solutions. Solving becomes a plain old fashioned evaluation of radical expressions, which does, in fact, actually involve *some* numerical proceedures in itself, but at a much later stage in the computation, like when evaluating the final numerical radical forms, such as $\sqrt{2}$. But, the latter is usually at or near the last step of the calculation. Closed form solutions essentially enable us to postpone such essential numerical computations to the last stages of a calculation proceedure, and avoid any such numerical considerations during the more intermediate steps. But here, without our `magic factor` method, we have to employ a numerical calculation rather early in our computation proceedure, to find the appropriate magnitudes, λ, before we can then use the closed form formulas for the directions to complete the method, viz., $q = \lambda \cdot p(\lambda)$.

When, $|p(\lambda)| \leq 1$, at the hill top, where, $\lambda = \sqrt{|c|/|a|}$, this means that, $|p(\lambda)| \leq 1$, $\forall\ \lambda \in [0, +\infty)$, so the whole curve, $f(\lambda)$, falls below the axis. Then, we can apply our `magic factor` approach, that immediately calculates the solutions. So, we divide our method into the two cases, `I:` $|p| \leq 1$, and, `II:` $|p| > 1$, where, $|p|$, is evaluated at that critical point, which is the location of the hill top, and we apply the different approaches required to find the solutions in each case. This works fine, and gives us the solutions. But, we've also seen that when we define the quadratic equation, $aq^2 + \beta \cdot bq + c = 0$, with an extra scalar factor, $\beta \in \mathbb{R}$, we can move that hill top up and down *continuously*, by increasing and decreasing the β parameter in a continuous manner. Now we recall, from complex analysis, that at times when we have a complex variable formula defined in some restricted region of the complex plane, that by a procedure, referred to as *analytic continuation*, that formula can often be extended to apply to regions outside of the initially defined restricted zone. This leads us to ask whether a similar thing might apply here, in the quaternion formulas. Can we extend the `magic factor` formulas, to apply to situations where the hill top rises above the axis? After all, we can get there by a continuous transformation. So, maybe there's a way to continuously transform the `magic factor` formulas to enable their use outside the restricted domain defined by the limitation, $|p(\lambda)| \leq 1$, $\forall \lambda \in [0, +\infty)$. If we succeeded in doing this, we could then dispense with the numerical search, such as using the MATLAB `fzero()` function call, altogether, and write everthing more traditionally in expressions with radicals.

But remember, the condition, $|p| < 1$, is used to derive the magic factors. So, we shouldn't necessarily expect things to work when, $|p| > 1$. And we already know that we can't even evaluate, $|p| > 1$, properly, to obtain the required finite measure, $|p| < \infty$, in those "*deep space*" equations where the curve has a natural spike. However, let's see what happens, if we just *naively* apply the `magic factor` method to the situation when the curve rises above the axis, but still has the "*hill top*" form, so that, at least, the provisional unit direction parameter, $|p|$, is finite there.

continue with this magic factor analysis...(working paper 01 – work in progress part)

`Dual Linear Equation.` Because the quadratic equation can be re-expressed in another form, we can use this other form to eliminate the specifically quadratic term, aq^2, and so obtain a purely '*linear equation*' in one unknown!

$$\underline{aq^2 + bq + c = 0} \tag{1.788}$$

$$q^2 + a^{-1}bq + a^{-1}c = 0 \tag{1.789}$$
$$q^2 - Q_1 q - qQ_2 + Q_1 Q_2 = 0 \qquad \leftarrow \quad (q - Q_1)(q - Q_2) = 0 \tag{1.790}$$
$$q^2 - Q_2 q - qQ_1 + Q_2 Q_1 = 0 \qquad \leftarrow \quad (q - Q_2)(q - Q_1) = 0 \tag{1.791}$$

$$(a^{-1}b + Q_1)q + qQ_2 + (a^{-1}c - Q_1 Q_2) = 0 \tag{1.792}$$

$$\text{Let, } \; A = (a^{-1}b + Q_1), \;\; B = Q_2, \;\; C = (a^{-1}c - Q_1 Q_2)$$

$$\underline{Aq + qB + C = 0} \;\; , \qquad \leftarrow \textit{ A linear equation dual to the quadratic!} \tag{1.793}$$

$$(a^{-1}b + Q_1)Q_1 + Q_1 Q_2 + (a^{-1}c - Q_1 Q_2) = 0 \;\; , \qquad q = Q_1, \;\; \checkmark \tag{1.794}$$
$$(a^{-1}b + Q_1)Q_2 + Q_2 Q_2 + (a^{-1}c - Q_1 Q_2) = 0 \;\; , \qquad q = Q_2, \;\; \checkmark \tag{1.795}$$

Here are three quadratic equations, expressing two different *forms*, and *a linear equation* in one variable, that all have the same two solutions, $q = Q_1$, and, $q = Q_2$. `A linear equation in one variable with two solutions!` Imagine that. How unexpected! Only in non-abelian algebra. If we tried this with abelian algebra, the product of the roots is the constant, $Q_1 Q_2 = a^{-1}c$, so, $C = (a^{-1}c - Q_1 Q_2) \equiv 0$, and the commuting property then yields, $Aq + qB = (A + B)q$. Then, $A + B = (a^{-1}b + Q_1 + Q_2)$, but, for abelian quadratic the negative sum of the roots is just the linear factor, $a^{-1}b = -(Q_1 + Q_2)$. So, we end up with, $0 \cdot q + 0 = 0$, $\forall q$! Any q can satisfy the linear form in abelian algebra. But move over to non-abelian algebra, and there are now two distinct roots to the linear equation!

Let's look at one of our previous numerical examples. We convert our previous quadratic equation into linear form.

$$(2 + \boldsymbol{i} - \boldsymbol{k})\boldsymbol{q}^2 + (-3 + \boldsymbol{j} + \boldsymbol{k})\boldsymbol{q} + (1 - \boldsymbol{j} + \boldsymbol{k}) = 0 \tag{1.796}$$

$$(-1.0793 + 0.4829\boldsymbol{i} + 0.1721\boldsymbol{j} + 0.1238\boldsymbol{k})\boldsymbol{q} + \boldsymbol{q}(1.0793 - 0.1014\boldsymbol{i} - 0.4773\boldsymbol{j} - 0.2001\boldsymbol{k}) + (0.1223 - 0.4364\boldsymbol{i} + 0.2453\boldsymbol{j} + 0.2954\boldsymbol{k}) = 0 \tag{1.797}$$

SOLUTIONS:

$$q1 = 0.0873 + 0.1496\boldsymbol{i} - 0.3279\boldsymbol{j} + 0.4571\boldsymbol{k} \tag{1.798}$$
$$q2 = 1.0793 - 0.1014\boldsymbol{i} - 0.4773\boldsymbol{j} - 0.2001\boldsymbol{k} \tag{1.799}$$

Plug any of the two solutions, q_1, q_2, shown, into either the quadratic or linear equation, and we can verify that both, q_1 and q_2, reduce the algebraic expressions to 0. Expected, for the quadratic, but a bit surprising for the linear.

Yet, we know how to solve linear equations. We can just use our new two-hand method. That method gives us *the unique solution*. Unique, meaning just one! Now, we're curious. Which solution, $q = Q_1$, or, $q = Q_2$, will our two-hand algebra for linear equations find? Well, as it turns out, the answer is "neither." Which, probably, isn't so surprising. And a bit fortunate. Otherwise we'd have a contradiction somewhere! There are some linear equations that can't be solved with our two-hand linear algebra, because they produce vanishing terms, like vanishing denominator, equivalent to zero determinant when converted to matrix form. The particular values contained in the three parameters, A, B, C, cause this equation to fall into that category, of indeterminate equations when tackled by our previous method. What is really amazing, is that obviously, some of those indeterminate equations have distinct solutions, that can be found by solving a quadratic equation instead. Just think about this for a moment. Some linear equations can only be solved by solving a corresponding quadratic equation. Once we find the right *dual quadratic equation*, we can then solve the linear equation.

Of course, the idea of "*distinct two*" for solutions, is probably just our imagination here. Since, we could always then combine them *linearly*, to obtain an infinite number of solutions, $q_{\alpha,\beta} = \alpha Q_1 + \beta Q_2, \;\; \alpha, \beta \in \mathbb{R}$, for the linear equation; something we can't do for the quadratic. Perhaps, a better label would be "*principal two*" roots.

In abelian algebra the two roots uniquely determine the quadratic equation, up to a scale factor. But, in quaternion algebra this is not generally the case.

$$\begin{aligned} f(q) &= a^{-1}[aq^2 + bq + c] - [(q - q_1)(q - q_2)] && (1.800)\\ &= a^{-1}bq + a^{-1}c + q_1 q + q q_2 - q_1 q_2 && (1.801)\\ &= (a^{-1}b + q_1)q + q q_2 + (a^{-1}c - q_1 q_2) && (1.802) \end{aligned}$$

At the two roots, q_1 and q_2, the function, $f(q)$, vanishes. But, otherwise this fuction has non-zero values. Discuss!.

Issues and Clarifications.

Before wrapping up this section #1, we need to discuss three major issues that appear in our results above, including some clarification on how we deal with the few ambiguous solutions.

`Issue #1:` The paradox of the unit assumption, $|p| = 1$.
`Issue #2:` The two roots to infinity paradox.
`Issue #3:` The "`Case 0`" solutions.

Recall that in real variable algebra, we start by making an assumption that the usual quadratic equation, $ax^2 + bx + c = 0,\ a, b, c, x \in \mathbb{R}$, has some solution. Then, we proceed to operate on this abelian quadratic expression using the rules of algebra, and we end up with a formula, $x = (-b \pm \sqrt{b^2 - 4ac})/(2a)$, that describes the "potential solution." But, then on checking, we find that in some cases, i.e., specifically, $b^2 - 4ac < 0$, the formula doesn't actually represent a solution in real numbers. So the "potential solution" formula, only works out some of the time. At other times, it produces a "non-solution" result. We don't usually think of this as a problem, that our initial assumption that there's a solution is contradicted by finding that sometimes there isn't one; Or, that the formula that is derived by the process, produces a non-solution instead. That's because, while our assumption, of existence of solution, motivated our search, and got us started on the path of derivation, the actual application of the rules of algebra did not depend on the assumption. So, finding there isn't a solution is not seen to be contradictory.

`Issue #1:` The paradox of the unit assumption, $|p| = 1$.

In our non-abelian quaternion equation, however, we begin with the assumption, $q = \lambda \cdot p,\ p, q \in \mathbb{H},\ \lambda \in \mathbb{R},\ \lambda > 0$, and we specifically require, $|p| = 1$, to enable us to justify using the then equivalent expression, $p^* = 1/p$, in the method of derivation. But, after applying the rules of algebra, some of which involve novel two-hand algebra methods, we end up with a formula for p, which clearly, in some cases, must have, $|p| < 1$, unquestionably contradicting the initial hypothesis, that, $|p| = 1$. The special case, $|p| = 0$, we can exclude, by specifically requiring, $b \neq 0$, in the given quadratic equation, $aq^2 + bq + c = 0$. But, we still end up with a whole range of solutions, $0 < |p| < 1$, for which the formula derived, $p(\lambda)$, can provide us with the solution when modified by what we call "magic factors," yet, the method of derivation itself is now suspect, since the resulting formula, for $p(\lambda)$, doesn't agree with the initial hypothesis on which its derivation depends! This seeming contradiction, is the first paradox. The formula works, we can use it to construct the solution, but how we got it currently rests on somewhat shakey grounds.

The way we deal with this above, is to separate the "method of derivation" from the "proof of solution," and to provide addtional proof that the provisional unit direction function, $p(\lambda)$, does actually produce the solution. Normally, the steps of derivation of a formula would be considered sufficient proof of the result, given that all regular well established rules of algebra are being applied. But, in this case, owing to the novel nature of the techniques used, and in consideration of the obvious paradoxes that appear, additional proving was warranted. So, we include the three cases, "Case 1: *Same magnitudes, different directions*," "Case 2: *Same directions, different magnitudes*," and, "Case 3: *Different magnitudes, different directions*," each of which involves substantial manipulation, in the quaternion algebra, to verify that the roots can indeed be found with the provisional unit direction formula, $p(\lambda)$.

`Issue #2:` The two roots to infinity paradox.

The second, somewhat troubling issue, is our intervening assumption that when we find, $|p(\lambda)| < 1,\ \forall \lambda \in (0, +\infty)$, that this means the provisional unit is actually an *average* of just "two" units, $p = (u_1 + u_2)/2$, which hypothesis is then used to artfully derive the "magic factors," enabling us to re-extract the two distinct directions, u_1 and u_2, from their average. The method works phenomenally well, except that in one case we find an "infinite" number of roots, seeming to contradict again, our initial hypothesis that "two" roots make up the average, and putting into question once more, the validity of our method of derivation. This is our second paradox. In this case, however, on proving the solution valid, we encounter a special formula that seems to explain how this is possible, and our paradox may melt away with time and better understanding of the unfolding algebra. On proof, we obtain the following result.

A new kind of average:

$$p = \frac{u_1 + u_2}{2} + \frac{u_1 - u_2}{2|u_1 - u_2|^2} \cdot [u_1, u_2]\ , \quad \forall\ u_1, u_2 \in \mathbb{H},\ |u_1|^2 = 1, |u_2|^2 = 1, u_1 \neq u_2, S(u_1) = S(u_2) = u_{00} \tag{1.803}$$

This remarkable formula, takes as input, any two distinct unit numbers, u_1, and, u_2, from the same infinite set that share equal scalar parts, and outputs the identical result each time, regardless of which two particular units are selected. The computed result is always the same as the *average* of two special units, $p = (u_1 + u_2)/2$, from that set, with exactly opposite vector parts, $V(u_1) = -V(u_2)$, which also means that the average, of the roots here, is a scalar, $p(\lambda = m) = u_{00} \in \mathbb{R}$. In this way, the formula achieves a consistent and unique average measure of a pair of unit directions among the infinite collection , being indifferent as to which pair is selected from that infinity! This, in turn, suggests that the formula may be used to define what is meant by the average of an infinite set of roots.

`Issue #3:` The "`Case 0`" solutions: convention: if there is at least 1 soln, then 2 solns reported with $|q_1| \leq |q_2|$.

We've given the solutions for, $a^*c > 0$, $a^*c < 0$, and, $w > 0$, $w < 0$, but what to do about, $a^*c = 0$, and, $w = 0$?

$$aq^2 + bq + c = 0\,, \qquad w = \lambda^2 a^*(a(\lambda p)^2 + b(\lambda p) + c)\,,\ \ \lambda = \sqrt{|c|/|a|} \tag{1.804}$$

`Case 0:` $a^*c = 0$ `out of scope` (1.805)

$b = 0,\ a^*c \in \mathbb{R},\ a^*c = 0$ (1.806)

$a^*c > 0:$ is "infinite # of roots," all with "same magnitudes, but different directions" (1.807)

$a^*c < 0:$ is "two roots," with "same magnitude, opposite directions" (1.808)

$a^*c = 0:$ is "all roots," with "same magnitude, same directions" – But, how many roots? (1.809)

One root? Two roots? Or, infinitely many 'equal' roots? our `convention`: "two roots" "$q_1 = q_2$" (1.810)

then, either, $a = 0$, or, $c = 0$, or, $a = c = 0$. (1.811)

1) $a = c = 0$, (1.812)

$0 \cdot q^2 + 0 \cdot q + 0 = 0\,,$ "any q" works to solve this eqn. our `convention`: "$q_1 = q_2 =$ `Inf`". (1.813)

2) $a = 0, c \neq 0$, (1.814)

$0 \cdot q^2 + 0 \cdot q + c = 0\,,$ "no q" works to solve this eqn. our `convention`: "$q_1 = q_2 =$ `NaN`". (1.815)

3) $a \neq 0, c = 0$, (1.816)

$a \cdot q^2 + 0 \cdot q + 0 = 0\,,$ "only q = 0" works to solve this eqn. our `convention`: "$q_1 = q_2 = 0$". (1.817)

`Case 0:` $a^*c = 0$, or, $w = 0$ `within scope` (1.818)

$b \neq 0,\ a^*c = 0,$ (1.819)

then, either, $a = 0$, or, $c = 0$, or, $a = c = 0$. (1.820)

1) $a = c = 0$, (1.821)

$0 \cdot q^2 + b \cdot q + 0 = 0\,,$ "only q = 0" works to solve this eqn. our `convention`: "$q_1 = 0,\ q_2 =$ Inf." (1.822)

2) $a = 0, c \neq 0$, (1.823)

$0 \cdot q^2 + b \cdot q + c = 0\,,$ "only $q = -b^{-1}c$" works to solve this eqn. our `convention`: "$q_1 = -b^{-1}c,\ q_2 =$ Inf." (1.824)

3) $a \neq 0, c = 0$, (1.825)

$a \cdot q^2 + b \cdot q + 0 = 0\,,$ "q = 0, $q = -a^{-1}b$" work to solve this eqn. well defined: "$q_1 = 0,\ q_2 = -a^{-1}b$." (1.826)

$b \neq 0,\ w \in \mathbb{R},\ w = 0$ (1.827)

$w > 0$ is "infinite # of roots," all have "same magnitude, but different directions" (1.828)

$w < 0$ is "two roots," with "same magnitude, different directions" (1.829)

$w = 0,$ is "all roots," with "same magnitude, same directions" – But, how many roots? (1.830)

One root? Two roots? Or, infinitely many 'equal' roots? our `convention`: "two roots" "$q_1 = q_2$" (1.831)

then, $|p| \leq 1$, and either, $p = 0$, or, $p \neq 0$ (1.832)

1) $w = 0,\ p = 0,\ a = c = 0.$ (1.833)

$0 \cdot q^2 + b \cdot q + 0 = 0\,,$ dealt with elsewhere, see above, a = c = 0, "within scope" decision flow control (1.834)

2) $w = 0,\ p = 0,\ a \neq 0,\ c = 0.$ (1.835)

$a \cdot q^2 + b \cdot q + 0 = 0\,,$ dealt with elsewhere, see above, $a \neq 0, c = 0$, "within scope" decision flow control (1.836)

3) $w = 0,\ p \neq 0,\ a \neq 0,\ c \neq 0,$ (1.837)

$a(\lambda p)^2 + b(\lambda p) + c = 0\,,$ "only $q = \lambda p$" works to solve this eqn. our `convention`: "$q_1 = \lambda p,\ q_2 = \lambda p$." (1.838)

These are our "`Case 0`" solutions. The, $a^*c = 0, w = 0$, solutions are at the boundary between the "finite" and the "infinite" sets of roots, and can be either; when equal, consistent with complex variables, by convention, we pick two equal roots; otherwise, we pick roots such that $|q_1| \leq |q_2|$, only returning `NaN` when no q exists.

#1-Reference: – the complete solution (formulae).

$$aq^2+bq+c=0 \qquad a,b,c,q\in\mathbb{H}_R \tag{1.839}$$

$$n_1,n_2,n_3\in\mathbb{R},\ n_1^2+n_2^2+n_3^2=1 \tag{1.840}$$

$$b=0: \tag{1.841}$$

$$aq^2+c=0 \tag{1.842}$$

$$\texttt{Case 1: } a^*c\in\mathbb{R},\ a^*c>0.: \qquad q=\ \sqrt{|c|/|a|}\cdot(n_1i+n_2j+n_3k),\ \ \texttt{``out of scope''}\ \infty\,\texttt{\#roots!} \tag{1.843}$$

$$\texttt{Case 2: } a^*c\in\mathbb{R},\ a^*c<0.: \qquad q=\pm\sqrt{|c|/|a|}\cdot 1 \tag{1.844}$$

$$\texttt{Case 3: } a^*c\notin\mathbb{R},\ S(a^*c)=0.: \quad q=\pm\sqrt{|c|/|a|}\cdot(1-a^*c/|a^*c|)/\sqrt{2} \tag{1.845}$$

$$\texttt{Case 4: } a^*c\notin\mathbb{R},\ S(a^*c)\neq 0.: \quad q=\pm\sqrt{|c|/|a|}\cdot(1-a^*c/|a^*c|/)/\sqrt{2-(a^*c+(a^*c)^*)/|a^*c|} \tag{1.846}$$

$$b\neq 0: \ \ |p(\lambda=\sqrt{|c|/|a|}\ \pm\ \epsilon)|>1,\ \ \epsilon\sim 10^{-5} \tag{1.847}$$

$$p=p(\lambda)=\frac{\lambda^5(\lambda^4a^*aa^*-c^*ac^*)(cb^*-\lambda^2ba^*)a+\lambda(c^*cc^*-\lambda^4a^*ca^*)(cb^*-\lambda^2ba^*)c}{(\lambda^4|a|^2-|c|^2)\cdot|\lambda^2a-c|^2\cdot|\lambda^2a+c|^2} \tag{1.848}$$

$$|p(\lambda)|=1,\ q=\lambda\cdot p(\lambda) \tag{1.849}$$

$$b\neq 0: \ \ |p(\lambda=\sqrt{|c|/|a}\ \pm\ \epsilon)|\leq 1,\ \ p=(a^*c\in\mathbb{R},a^*c>0)\ ?\ P_3\ :\ P_1;\ \texttt{L'Hôspital's Rule}^{[\dagger]} \tag{1.850}$$

$$P_1(\lambda=\sqrt{|c|/|a|})=\frac{1}{\sqrt{|c|/|a|}}\cdot\frac{1}{4}\left[\ \frac{(\ 5a^*c-3|a^*c|\ )b^*a\ \ +\ \ |a|^2(\ 1-3a^*c/|a^*c|\ )b^*c}{|a|^2(\ 2|a^*c|-(a^*c+(a^*c)^*)\ )}\ \ -\ \ \frac{5a^*b}{|a|^2}\ \right] \tag{1.851}$$

$$P_3(\lambda=\sqrt{|c|/|a|})=\frac{1}{\sqrt{|c|/|a|}}\cdot\left[\ \frac{b^*a-2a^*b}{2|a|^2}\ \right] \tag{1.852}$$

Onplane Magic Factors: $|p(\lambda)|<1,\ \ \lambda=\sqrt{|c|/|a|},\ \ a,b,c\in\mathbb{C}\subset\mathbb{H}_R$ (1.853)

$$q_1\ \ =\ \ \lambda\cdot u_1\ \ =\ \ \lambda\cdot p\cdot\left(\ 1+\frac{(p-p^*)}{|p-p^*|}\sqrt{|p|^{-2}-1}\ \right) \tag{1.854}$$

(1.855)

$$q_2\ \ =\ \ \lambda\cdot u_2\ \ =\ \ \lambda\cdot p\cdot\left(\ 1-\frac{(p-p^*)}{|p-p^*|}\sqrt{|p|^{-2}-1}\ \right)$$

Offplane Magic Factors: $|p(\lambda)|<1,\ \ \lambda=\sqrt{|c|/|a|},\ \ a,b,c\in\mathbb{H}_R$ (1.856)

$$q_1=\lambda\cdot u_1=\lambda\cdot\left(p-u\cdot\sqrt{\frac{|a(\lambda p)^2+b(\lambda p)+c|}{|c|}}\right) \tag{1.857}$$

$$q_2=\lambda\cdot u_2=\lambda\cdot\left(p+u\cdot\sqrt{\frac{|a(\lambda p)^2+b(\lambda p)+c|}{|c|}}\right) \tag{1.858}$$

$$w=\lambda^2a^*(a(\lambda p)^2+b(\lambda p)+c)\ \ ,\qquad \lambda=\sqrt{\frac{|c|}{|a|}} \tag{1.859}$$

where, $u=u(w)$: (1.860)

$$\texttt{Case 1: } w\in\mathbb{R},\ w>0.: \qquad u=\ (n_1i+n_2j+n_3k),\ \ \texttt{``within scope''}\ \infty\,\texttt{\#roots!} \tag{1.861}$$

$$\texttt{Case 2: } w\in\mathbb{R},\ w<0.: \qquad u=\ 1 \tag{1.862}$$

$$\texttt{Case 3: } w\notin\mathbb{R},\ S(w)=0.: \quad u=\ (1-w/|w|)/\sqrt{2} \tag{1.863}$$

$$\texttt{Case 4: } w\notin\mathbb{R},\ S(w)\neq 0.: \quad u=\ (1-w/|w|/)/\sqrt{2-(w+w^*)/|w|} \tag{1.864}$$

[†] The construct, $p=(X)\ ?\ Y:Z$; follows the C-language inline if-then-else assignment clause.

#1-Reference: – the complete solution (code).

A complete hypercomplex number quadratic equation solver to: $ah^2 + bh + c = 0$, for reals, $h = x \in \mathbb{R}$, complex numbers, $h = z \in \mathbb{C}$, and quaternions, $h = q \in \mathbb{H}$. in MATLAB code:

```
function [Q1 Q2]=aqqbqc(A,B,C)
% Quaternion Quadratic Equation Solver   ver:  1.0    (aqq + bq + c = 0)
global a b c
psolqabc(A,B,C)
QInf = [Inf, Inf, Inf, Inf];

if a == 0 % linear eqn.
% bq + c = 0
q = -b^(-1)*c;
Q1 = [ real(q(1,1)), imag(q(1,1)), real(q(1,2)), imag(q(1,2)) ];
Q2 = QInf; % recall:  q = [w,v;-v',w'], w = t + x*1i, v = y + z*1i
return;             % q = t + xi + yj + zk
end
if c == 0 % linear eqn, + 0
% (aq + b)q = 0
q = -a^(-1)*b;
Q1 = [ 0, 0, 0, 0]; % by convention |Q1| <= |Q2| !
Q2 = [ real(q(1,1)), imag(q(1,1)), real(q(1,2)), imag(q(1,2)) ];
return;
end
r = sqrt(sqrt(trace(c*c')/trace(a*a')));
ac = a'*c;
acImagTest = abs(imag(ac(1,1))) + abs(real(ac(1,2))) + abs(imag(ac(1,2)));
% = abs(x) + abs(y) + abs(z); t + xi + yj + zk

if b == 0 % one-step methods
% aq^2 + c = 0
if acImagTest == 0 % i.e.  a*.c in [R]
   if real(ac(1,1)) > 0 % i.e.  a*.c > 0
    % Case 1:  "out of scope" infinite # of roots
    % pick an arbitary pair of valid pure quaternion roots
    % q = r*[0+1i,1+1i;-1+1i,0-1i]/sqrt(3);
    % =or= pick a 'random' opposite pair of valid pure quaternion roots
    n = -1 + 2*rand([1,3]); n = n/sqrt(n*n');
    q = r*[0+n(1)*1i,n(2)+n(3)*1i;-n(2)+n(3)*1i,0-n(1)*1i];
   else % i.e.  a*.c < 0
    % Case 2:  real roots
    q = r*[1,0;0,1];
   end
else % a*.c is in [H]
   q = r*([1,0;0,1] - ac/sqrt(abs(trace(ac*ac'))/2));
   if real(ac(1,1)) == 0
    % Case 3:  S(a*.c) = 0
    q = q/sqrt(2);
   else
    % Case 4:  S(a*.c) != 0
    q = q/sqrt(2 - 2*real(ac(1,1))/sqrt(abs(trace(ac*ac'))/2));
   end
end
Q1 = [ real(q(1,1)), imag(q(1,1)), real(q(1,2)), imag(q(1,2)) ];
Q2 = -Q1;
return;
end % .of b == 0 cases.
```

```
% b != 0 cases :  two-step methods
p = psolqv(r-10^-5);
pp = p*p';
if pp > 1
   r1 = fzero(@psolq,[0,r-10^-5]);
   r2 = fzero(@psolq,[r+10^-5,10^23]);
   p1 = psolqv(r1);
   p2 = psolqv(r2);
else
   % offplane magic factor solution method 'third step'
   P = psolqvh0; % L'Hospital's rule evaluate P=p(r=sqrt(|c|/|a|))
   Q = r*P; % avg.  root
   E = mulq(A,mulq(Q,Q)) + mulq(B,Q) + C; % quadratic residual
   cA = -A; cA(1) = A(1); % conjugate of A
   W = r^2*mulq(cA,E); % weighted residual quaternion
   wImagTest = abs(W(2)) + abs(W(3)) + abs(W(4));
   % = abs(x) + abs(y) + abs(z) ; t + xi + yj + zk
   if wImagTest == 0 % W in [R]
    if W(1) == 0
     % Case 0:  soln found !  two roots both equal to avg root!
     Q1 = Q; Q2 = Q1;
     return;
    elseif W(1) > 0
     % Case 1:  W in [R], W > 0.
     % "within scope" infinite # of roots
     % pick arbitrary valid pure quaternion unit for u
     % U = [0,1,1,1]/sqrt(3);
     % =or= pick a 'random' valid pure quaternion unit
     n = -1 + 2*rand([1,3]); n = n/sqrt(n*n');
     U = [0,n(1),n(2),n(3)];
    else % W(1) < 0
     % Case 2:  W in [R], W < 0.
     U = [1,0,0,0];
    end
   else % W in [H]
    U = ([1,0,0,0] - W/sqrt(W*W'));
    if W(1) == 0
     % Case 3:  W in [H], S(W) = 0.
     U = U/sqrt(2);
    else
     % Case 4:  W in [H], S(W) != 0.
     U = U/sqrt(2 - 2*W(1,1)/sqrt(W*W'));
    end
   end
   % calc the magic !
   fixup = sqrt(sqrt((E*E')/(C*C')));
   p1 = P - U*fixup;
   p2 = P + U*fixup;
   r1 = r; r2 = r;
end
Q1 = r1*p1;
Q2 = r2*p2;
end % .of fn aqqbqc()
```

Note: In the case of an infinte number of root solutions, the function, `[Q1 Q2]=aqqbqc(A,B,C)` , selects a random pair of valid roots, out of that infinite set, and returns these in `Q1, Q2`. Therefore, calling the function two or more times, with same input parameters, will result in different solution pairs. In this way, infinity can be determined.

#1-Reference: – the complete solution (formulae miscellaneous index).

unit magnitude solution: pg.2

$$q = \frac{(a^*aa^* - c^*ac^*)(cb^*a - ba^*a) + (c^*cc^* - a^*ca^*)(cb^*c - ba^*c)}{(|a|^2 - |c|^2)((|a|^2 + |c|^2)^2 - ((ac^*) + (ac^*)^*)^2)} \tag{1.14}$$

provisional unit direction function - standard form: pg.2

$$p = \frac{(\lambda^6|a|^2a^* - \lambda^2c^*ac^*)(\lambda^3cb^*a - \lambda^5ba^*a) + (|c|^2c^* - \lambda^4a^*ca^*)(\lambda cb^*c - \lambda^3ba^*c)}{(\lambda^4|a|^2 - |c|^2)((\lambda^4|a|^2 + |c|^2)^2 - \lambda^4((ac^*) + (ac^*)^*)^2)} \tag{1.16}$$

provisional unit direction function - implemented form: pg.8

$$p = \frac{\lambda^5(\lambda^4a^*aa^* - c^*ac^*)(cb^* - \lambda^2ba^*)a + \lambda(c^*cc^* - \lambda^4a^*ca^*)(cb^* - \lambda^2ba^*)c}{(\lambda^4|a|^2 - |c|^2) \cdot |\lambda^2a - c|^2 \cdot |\lambda^2a + c|^2} \tag{1.57}$$

provisional unit direction function - 1st optimized form: pg.71

$$p = \frac{\lambda(\lambda^2a^* - c^*)(cb^*(\lambda^2a - c) - \lambda^2(\lambda^2a - c)a^*b)}{(\lambda^4|a|^2 - |c|^2)|\lambda^2a - c|^2} \tag{1.683}$$

provisional unit direction function - 2nd optimized form: pg.71

$$p = \frac{\lambda(\lambda^2a^* - c^*)cb^*(\lambda^2a^* - c^*)^{-1} - \lambda^3a^*b}{(\lambda^4|a|^2 - |c|^2)} \tag{1.684}$$

provisional unit direction function - lead reduced implemented form: pg.52

$$p = \frac{\lambda^5(\lambda^4 - c^*c^*)(cb^* - \lambda^2b) + \lambda(c^*cc^* - \lambda^4c)(cb^* - \lambda^2b)c}{(\lambda^4 - |c|^2) \cdot |\lambda^2 - c|^2 \cdot |\lambda^2 + c|^2} \tag{1.408}$$

provisional unit direction function - lead reduced 1st optimized form: pg.53

$$p = \lambda \cdot (\lambda^2 - c^*) \cdot \left(\frac{cb^* \cdot (\lambda^2 - c) - \lambda^2(\lambda^2 - c) \cdot b}{(\lambda^4 - |c|^2) \cdot |\lambda^2 - c|^2}\right) \tag{1.414}$$

provisional unit direction function - lead reduced 2nd optimized form: pg.72

$$p = \frac{\lambda(\lambda^2 - c^*)cb^*(\lambda^2 - c^*)^{-1} - \lambda^3b}{(\lambda^4 - |c|^2)} \tag{1.685}$$

provisional unit direction function = abelian + non-abelian expansion: pg.47

$$p(\lambda) = \left(\frac{m\lambda}{\lambda^2 + m^2}\right) \cdot s$$

$$+ \left(\frac{m\lambda}{\lambda^2 + m^2}\right) \cdot \left[1 - \left(\frac{(\lambda^4d^2 - m^4(d^*)^2)(\lambda^2m^2d^* - m^4d)}{|(\lambda^4d^2 - m^4(d^*)^2)|^2}\right) \cdot (d^* + d)\right] \cdot \frac{d \cdot [d, s]}{2|d|^2} \tag{1.335}$$

for the case of same magnitudes, different directions:

$q_1 = m \cdot u_1$, $q_2 = m \cdot u_2$, $s = u_1 + u_2$, $d = u_1 - u_2$.

#1-Reference: – the complete solution (formulae miscellaneous index cont'd.).

"width" formulas for the "*difference*" and "*sum*" of magnitudes: $(\pm \to +)$ pg.64,63

$$w = \lambda_2 - \lambda_1 \ , \quad W = \lambda_1 + \lambda_2 \ , \quad q_1 = \lambda_1 \cdot u_1, \ q_2 = \lambda_2 \cdot u_2, \ \ \lambda_2 > \lambda_1 \tag{1.633}$$

$$w^2 = \frac{[|b|^2 - 4|a^*c|] \pm \sqrt{(|b|^2 - 4|a^*c|)^2 - 4(-|a^*c||b|^2[2 - (u_1u_2^* + u_2u_1^*)] + |a^*c|^2[2 - (u_1^2u_2^{*2} + u_2^2u_1^{*2})])}}{2|a|^2} \tag{1.612}$$

$$W^2 = \frac{[|b|^2 + 4|a^*c|] \pm \sqrt{(|b|^2 - 4|a^*c|)^2 - 4(-|a^*c||b|^2[2 - (u_1u_2^* + u_2u_1^*)] + |a^*c|^2[2 - (u_1^2u_2^{*2} + u_2^2u_1^{*2})])}}{2|a|^2} \tag{1.623}$$

"*product*" of magnitudes: pg.60 $\quad \lambda_1\lambda_2 = \lambda_0^2 = |c|/|a|$ (1.589)

formal parameters, (a, b, c), expressed in terms of roots: pg.41,16

$$a = +a \ , \quad q_1 = m \cdot u_1 \ , \ \ q_2 = n \cdot u_2 \tag{1.199}$$

$$b = -a(m^2u_1^2 - n^2u_2^2)(mu_1^* - nu_2^*)/|mu_1 - nu_2|^2 \quad = \quad -a(q_1^2 - q_2^2)(q_1^* - q_2^*)/|q_1 - q_2|^2 \tag{1.96}$$

$$c = +amn(mu_1 - nu_2)u_2(mu_1^* - nu_2^*)u_1/|mu_1 - nu_2|^2 \quad = \quad +a(q_1 - q_2)q_2(q_1^* - q_2^*)q_1/|q_1 - q_2|^2 \tag{1.103}$$

lead reduced characteristic 6th degree polynomial in λ^2: pg.74

$$\lambda^{2\cdot 6} + \alpha_5\lambda^{2\cdot 5} + \alpha_4\lambda^{2\cdot 4} + \alpha_3\lambda^{2\cdot 3} + \alpha_2\lambda^{2\cdot 2} + \alpha_1\lambda^{2\cdot 1} + \alpha_0 = 0 \tag{1.711}$$

with coefficients:

$$\begin{aligned}
\alpha_5 &= -|b|^2 - (c + c^*) \\
\alpha_4 &= -|c|^2 + |b|^2(c + c^*) + (bc^*b + b^*cb^*) \\
\alpha_3 &= -2|c|^2(|b|^2 - (c + c^*)) - (bc^*b + b^*cb^*)(c + c^*) - [b, c]^2 \\
\alpha_2 &= [-|c|^2 + |b|^2(c + c^*) + (bc^*b + b^*cb^*)] \cdot |c|^2 \\
\alpha_1 &= [-|b|^2 - (c + c^*)] \cdot |c|^4 \\
\alpha_0 &= +|c|^6
\end{aligned} \tag{1.712}$$

evaluation of $p(\lambda)$ at critical point, $\lambda = \sqrt{|c|/|a|}$, by *L'Hôspital's Rule*: pg.70

$$P_1(\lambda = \sqrt{|c|/|a|}) = \frac{1}{\sqrt{|c|/|a|}} \cdot \frac{1}{4}\left[\frac{(\ 5a^*c - 3|a^*c|\ )b^*a \ \ + \ \ |a|^2(\ 1 - 3a^*c/|a^*c|\ )b^*c}{|a|^2(\ 2|a^*c| - (a^*c + (a^*c)^*)\ )} \ \ - \ \ \frac{5a^*b}{|a|^2} \right] \tag{1.678}$$

$$P_3(\lambda = \sqrt{|c|/|a|}) = \frac{1}{\sqrt{|c|/|a|}} \cdot \left[\frac{b^*a - 2a^*b}{2|a|^2} \right] \tag{1.680}$$

lead reduced evaluation of $p(\lambda)$ at critical point, $\lambda = \sqrt{|c|}$, by *L'Hôspital's Rule*: pg.70

$$P_1(\lambda = \sqrt{|c|}) = \frac{1}{\sqrt{|c|}} \cdot \frac{1}{4}\left[\frac{(\ 5c - 3|c|\ )b^* \ \ + \ \ (\ 1 - 3c/|c|\ )b^*c}{2|c| - (c + c^*)} \ \ - \ \ 5b \right] \tag{1.677}$$

$$P_3(\lambda = \sqrt{|c|}) = \frac{1}{\sqrt{|c|}} \cdot \left[\frac{1}{2}b^* - b \right] \tag{1.679}$$

#1-Reference: – the complete solution (partly optimized standalone code).

```
function [Q1 Q2]=aqqbqc(A,B,C) % by pmj
% Quaternion Quadratic Equation Solver ver:  1.1, (aqq + bq + c = 0)
QZro = [ 0, 0, 0, 0];
QInf = [Inf,Inf,Inf,Inf];
QNaN = [NaN,NaN,NaN,NaN];
vTOm = @(V) [V(1)+1i*V(2),V(3)+1i*V(4);-V(3)+1i*V(4),V(1)-1i*V(2)];
mTOv = @(m) [real(m(1,1)), imag(m(1,1)), real(m(1,2)), imag(m(1,2))];
a = vTOm(A); b = vTOm(B); c = vTOm(C);

if     ((a == 0) & (b == 0) & (c == 0)); Q1 = QInf; Q2 = QInf; return;
elseif ((a == 0) & (b == 0) & (c ~= 0)); Q1 = QNaN; Q2 = QNaN; return;
elseif ((a == 0) & (b ~= 0)); Q1 = mTOv(-(b^-1)*c); Q2 = QInf; return;
elseif ((a ~= 0) & (c == 0)); Q1 = QZro; Q2 = mTOv(-(a^-1)*b); return;
else % ((a ~= 0) & (c ~= 0));   % ** by convention |Q1| <= |Q2| **
I = [1,0;0,1]; b = (a^-1)*b; c = (a^-1)*c; % lead reduce params
r = sqrt(sqrt(trace(c*c')/2));
end

pm = @(r) ((r*(r^2*I-c')*c*b'*(r^2*I-c')^-1) - r^3*b)/(trace(r^4*I-c*c')/2);
pf = @(r) mTOv(pm(r))*mTOv(pm(r))' - 1;   % f(r) = |p(r)|^2 - 1
ImagTest = @(w) abs(imag(w(1,1))) + abs(real(w(1,2))) + abs(imag(w(1,2)));

P = mTOv(pm(r-10^-5)); PP = P*P';

if PP > 1
   r1 = fzero(pf,[0,r-10^-5]);
   r2 = r*r/r1;
   Q1 = r1 * mTOv(pm(r1));
   Q2 = r2 * mTOv(pm(r2));
   return
end
% PP <= 1
if b == 0
   p = [0,0;0,0];
else
   mc = sqrt(abs(trace(c*c')/2)); % mc = |c|
   if ImagTest(c) == 0 && real(c(1,1)) > 0 % use p = P3;
    p = (1/r)*( (1/2)*b' - b );
   else % use p = P1;
    p = (1/r)*(1/4)*(((5*c - 3*mc*I)*b' + (I - 3*c/mc)*b'*c)/(2*mc - 2*real(c(1,1))) - 5*b);
   end
end

q = r*p; Q = mTOv(q); % avg.  root
w = q*q + b*q + c; W = mTOv(w);   % quadratic residual weighted and simplified

if ImagTest(w) ~= 0;    U = ([1,0,0,0] - W/sqrt(W*W'))/sqrt(2 - 2*W(1)/sqrt(W*W'));
elseif W(1) < 0;        U = [1,0,0,0];
elseif W(1) == 0;       U = [0,0,0,0]; % equal roots
else % W(1) > 0; infinite # roots, pick valid pair at random!
   n = -1 + 2*rand([1,3]); n = n/sqrt(n*n'); U = [0,n];
end
U = U*sqrt(sqrt(W*W')); % fixup
Q1 = Q - U;
Q2 = Q + U;
end % .of fn aqqbqc()
```

#2. A second quadratic equation.

Consider the following quadratic equation ($a, b, c, q \in \mathbb{H}_R$),

$$q^2 a + qb + c = 0 \quad , \quad b \neq 0 \tag{2.1}$$

where, as before, a, b, and, c, are initially special parameters, that result in a unit quaternion, q, with, $|q| = 1$, being one solution. We proceed as before, dividing this time on the left, by, q. Our equation is just the complete reverse of the first equation previously considered. So, this time different parameters will be hand transformed.

$$\begin{aligned} qa &+ q^* c &&= -b && (2.2)\\ c^* q &+ a^* q^* &&= -b^* && (2.3) \end{aligned}$$

Converting, from Hamilton's 19th century $\mathbb{H}$-*algebra*, to our new 21st century $\mathbb{X}$-*algebra*, we obtain,

$$\begin{aligned} a'\hat{q} &+ c'\hat{q}^* &&= -\hat{b} && (2.4)\\ c^*\hat{q} &+ a^*\hat{q}^* &&= -\hat{b}^* && (2.5) \end{aligned}$$

Prepare the equations for elimination of the conjugate, $\hat{q}^*$, by muliplying each by suitable factors,

$$\begin{aligned} a^* a'\hat{q} &+ a^* c'\hat{q}^* &&= -a^*\hat{b} && (2.6)\\ c^* c'\hat{q} &+ a^* c'\hat{q}^* &&= -c'\hat{b}^* && (2.7) \end{aligned}$$

We like to keep the left handed factors to the right, and right handed factors to the left. It's just a convenient calculation convention. Not really a necessary thing, but useful in the art. We subtract the lower from upper to get our equation in just the one unknown, $\hat{q}$.

$$(a^* a' - c^* c')\hat{q} \quad = \quad (-a^*\hat{b} + c'\hat{b}^*) \tag{2.8}$$

Next, we multiply both sides by the right conjugate, $h^{*R} \equiv (a^* a' - c^* c')^{*R}$, factor,

$$\begin{aligned} (a^* a' - c^* c')^{*R}(a^* a' - c^* c')\hat{q} &= (a^* a' - c^* c')^{*R}(-a^*\hat{b} + c'\hat{b}^*) && (2.9)\\ (aa' - cc')(a^* a' - c^* c')\hat{q} &= (aa' - cc')(-a^*\hat{b} + c'\hat{b}^*) && (2.10) \end{aligned}$$

Again, this version of the procedure being outlined here is possible because, $h \equiv (a^* a' - c^* c')$, only has 2-terms. Other equations will require more advanced techniques, discussed later below. Multiplying out,

$$(|a|^2(a')^2 - ac^* a' c' - ca^* c' a' + |c|^2(c')^2)\hat{q} \quad = \quad (aa' - cc')(-a^*\hat{b} + c'\hat{b}^*) \tag{2.11}$$

We multiply both sides now, by the left-conjugate of the new factor, $(|a|^2(a')^2 - ac^* a' c' - ca^* c' a' + |c|^2(c')^2)^{*L}$,

$$\begin{aligned} &(|a|^2(a'^*)^2 - ac^* c'^* a'^* - ca^* a'^* c'^* + |c|^2(c'^*)^2) \cdot (|a|^2(a')^2 - ac^* a' c' - ca^* c' a' + |c|^2(c')^2)\hat{q} \\ &= \quad (|a|^2(a'^*)^2 - ac^* c'^* a'^* - ca^* a'^* c'^* + |c|^2(c'^*)^2) \cdot (aa' - cc')(-a^*\hat{b} + c'\hat{b}^*) \end{aligned} \tag{2.12}$$

multiplying out and rearranging,

$$\hat{q} = \left(\begin{array}{c} |a|^2(aa^* aa^{*\prime} - ac^* ac^{*\prime} - ca^{*\prime} c^{*\prime} a' - ca^{*\prime} a^{*\prime} c') \\ +|c|^2(ac^{*\prime} c^{*\prime} a' + ac^{*\prime} a^{*\prime} c' + ca^* ca^{*\prime} - cc^* cc^{*\prime}) \\ \hline (|a|^2 - |c|^2)^2((|a|^2 + |c|^2)^2 - ((ac^*) + (ac^*)^*)^2) \end{array} \right) \cdot (-a^*\hat{b} + c'\hat{b}^*) \tag{2.13}$$

simplifying,

$$q = \frac{(aa^* a - ca^* c)(b^* ca^* - a^* ba^*) + (cc^* c - ac^* a)(b^* cc^* - a^* bc^*)}{(|a|^2 - |c|^2)((|a|^2 + |c|^2)^2 - (ac^* + (ac^*)^*)^2)} \tag{2.14}$$

This is the "*unit magnitude*" solution, in the special case where, a, b, c, are such that the quadratic has at least one unit quaterion for a solution. We now relax that requirement, and let, a, b, c, take any values, and replace, $q = \lambda p$,

$$p^2\lambda^2 a + p\lambda b + c = 0 \tag{2.15}$$

This new quadratic has the same form as before, so has the same solution formula, with, p, being now the unit quaternion variable. So, to obtain the expression for the '*provisional*' unit quaternion, p, we make the replacements, $a \to \lambda^2 a,\ b \to \lambda b,\ c \to c$, to obtain,

$$p = \frac{(\lambda^6|a|^2 a - \lambda^2 ca^*c)(\lambda^3 b^* ca^* - \lambda^5 a^* ba^*) + (|c|^2 c - \lambda^4 ac^* a)(\lambda b^* cc^* - \lambda^3 a^* bc^*)}{(\lambda^4|a|^2 - |c|^2)((\lambda^4|a|^2 + |c|^2)^2 - \lambda^4(ac^* + (ac^*)^*)^2)} \tag{2.16}$$

Now we consider, p, to be a function of λ, and solve for λ in the constraint equation, $|p(\lambda)| = 1$, or equivalently,

$$p(\lambda)p(\lambda)^* = 1 \tag{2.17}$$

we can then write the solution to this second quadratic equation for unknown, q, in terms of each λ found that satisfies this constraint equation. Then, for each λ found, q, is given by,

$$q = \lambda \cdot \frac{(\lambda^6|a|^2 a - \lambda^2 ca^*c)(\lambda^3 b^* ca^* - \lambda^5 a^* ba^*) + (|c|^2 c - \lambda^4 ac^* a)(\lambda b^* cc^* - \lambda^3 a^* bc^*)}{(\lambda^4|a|^2 - |c|^2)((\lambda^4|a|^2 + |c|^2)^2 - \lambda^4(ac^* + (ac^*)^*)^2)} \tag{2.18}$$

$$= Tq \cdot Uq \tag{2.19}$$

where, Tq and Uq, are Hamilton's *tensor* and *versor* notation, the equivalent of *magnitude* and *direction*.

Example in Quaternions (#2)

In preparation for implementation, we rearrange the numerator and denominator in the provisional unit direction, $p(\lambda)$, re-writing it,

$$p = \frac{\lambda^5(\lambda^4 aa^* a - ca^* c)(b^* c - \lambda^2 a^* b)a^* + \lambda(cc^* c - \lambda^4 ac^* a)(b^* c - \lambda^2 a^* b)c^*}{(\lambda^4|a|^2 - |c|^2) \cdot |\lambda^2 a - c|^2 \cdot |\lambda^2 a + c|^2} \tag{2.20}$$

This formula is very similar to our previous result. The denominator is the same. While, the numerator is modified by changing the conjugation profile of the parameters, a, b, c, appearing in the expressions, and reversing a few products. We can reuse most of the same MATLAB code as before. We just need to revise the three functions, `psolqd(r)`, `psolqev(Q)`, and, `psolqvh(r)`, to take into account these changes. The first two replacement functions for 'ver.2' of the code are here; the last is given later in the section on "Vanishing Denominator."

in MATLAB code:

```
function p=psolqd(r)     % ver:  2.0,  (qqa + qb + c = 0)
global a b c;
np = r^5*(r^4*a*a'*a - c*a'*c)*(b'*c - r^2*a'*b)*a';
np = np + r*(c*c'*c - r^4*a*c'*a)*(b'*c - r^2*a'*b)*c';
dp = trace((r^4*a*a' - c*c')*(r^2*a - c)*(r^2*a - c)'*(r^2*a + c)*(r^2*a + c)')/2;
p = np/dp;
end

function V=psolqev(Q)     % ver:  2.0,  (qqa + qb + c = 0)
global a b c;
q = [(Q(1)+1i*Q(2)),(Q(3)+1i*Q(4));-(Q(3)+1i*Q(4))',(Q(1)+1i*Q(2))'];
v = q*q*a + q*b + c;
V = [real(v(1,1)) , imag(v(1,1)), real(v(1,2)), imag(v(1,2))];
end
```

When, $c = 0$, the formula reduces to, $p(\lambda) = -ba^*/(\lambda|a|^2)$. Requirement, $|p(\lambda)| = 1$, then gives, $\lambda = |b|/|a|$, which finds unit direction, $p = -ba^*/|ba|$, and final solution, $q = \lambda p = (|b|/|a|) \cdot (-ba^*/|ba|) = -ba^{-1}$. When, $a = 0$, formula reduces to, $p(\lambda) = -\lambda cb^*/|c|^2$. Requirement, $|p(\lambda)| = 1$, gives, $\lambda = |c|/|b|$, and thus, $p = -cb^*/|cb|$, then, solution, $q = \lambda p = (|c|/|b|) \cdot (-cb^*/|cb|) = -cb^{-1}$. The case, $b = 0$, for, $q^2 a + qb + c = 0$, is derived in APPENDIX C.

Equation Transformations:

$$\#1: \quad aq^2+bq+c=0 \tag{2.21}$$
$$\uparrow$$
$$q=Q+d \tag{2.22}$$
$$a(Q+d)^2+b(Q+d)+c=0 \tag{2.23}$$
$$a(Q^2+Qd+dQ+d^2)+bQ+bd+c=0 \tag{2.24}$$
$$aQ^2+aQd+adQ+ad^2+bQ+bd+c=0 \tag{2.25}$$
$$aQ^2+(ad+b)Q+aQd+ad^2+bd+c=0 \tag{2.26}$$
$$\text{set,} \quad ad+b=0 \tag{2.27}$$
$$d=-a^{-1}b \;\; = \;\; -a\backslash b \tag{2.28}$$
$$aQ^2+aQ(-a^{-1}b)+a(-a^{-1}b)(-a^{-1}b)+b(-a^{-1}b)+c=0 \tag{2.29}$$
$$aQ^2-aQa^{-1}b+ba^{-1}b-ba^{-1}b+c=0 \tag{2.30}$$
$$aQ^2-aQa^{-1}b+c=0 \tag{2.31}$$
$$Q^2-Qa^{-1}b+a^{-1}c=0 \tag{2.32}$$
$$Q^2a-Qa^{-1}ba+a^{-1}ca=0 \tag{2.33}$$
$$Q^2A+QB+C=0\;, \qquad \leftarrow \;\; A=a,\; B=-a^{-1}ba,\; C=a^{-1}ca \tag{2.34}$$
$$\downarrow$$
$$\#2: \quad q^2a+qb+c=0 \tag{2.35}$$

Translation swaps the action; a displacement, d, turns right action, bq, into left action, qb. The b and c are rotated about a, by a, to produce, $b \to -a^{-1}ba$, and, $c \to a^{-1}ca$. But note, *all* known parameters move to the other side. The whole equation is reversed! Our first quadratic, becomes this second quadratic, and visa versa, by a simple transformation on the unknown variable, q; that transformation here, required in this case, is a translation.

By its very nature, as a composite sequence of $+,-,\times,\div$, operations, the transformation here is reversible!

$$(a,b,c) \to (A,B,C) \quad \checkmark \tag{2.36}$$
$$(a,b,c) \leftarrow (A,B,C) \quad \checkmark \tag{2.37}$$

This means that we may also obtain the solution to the second quadratic equation, by simply transforming the solution to the first quadratic equation, or visa versa. If we have the solution, q, to the first equation, we can write the solution, Q, to the second equation, by putting in the displacement, $Q=q+a^{-1}b=q-BA^{-1}$; and if we have the solution, Q, to the second equation, we can write the solution, q, to the first equation, by putting in the *reverse* displacement, $q=Q-a^{-1}b=Q+BA^{-1}$.

Since, q, is a function of, a,b,c, i.e., $q=q(a,b,c)$, and, Q, is a function of, A,B,C, i.e., $Q=Q(A,B,C)$, we just need to substitute the known parameters in their transformed state into the solution formulas of the other, so,

$$q(a,b,c) \quad = Q(a,-a^{-1}ba,a^{-1}ca)-a^{-1}b \tag{2.38}$$

$$Q(A,B,C) \quad = q(A,-ABA^{-1},ACA^{-1})-BA^{-1} \tag{2.39}$$

So, now we have the option of using the two-hand solution to quadratic #2, $q^2a+qb+c=0$, replacing the MATLAB 'ver.1' code for the three functions, `psolqd(r)`, `psolqev(Q)`, and `psolqvh(r)`, with the 'ver.2' counterparts, given here, or we can simply use the old 'ver.1' code, as is, that solved quadratic #1, $aq^2+bq+c=0$, and just enter the transformed parameters, $A \to A$, $B \to -ABA^{-1}$, $C \to ACA^{-1}$, remembering to add back that displacement after the calculation is done; $Q=q-BA^{-1}$, to get the right final solutions to the *reversed* quadratic quation. To help with this alternative, we define two new convenient MATLAB functions, `mulq(A,B)` and `invq(A)`, that multiply and invert when quaternions are in the row vector format.

in MATLAB code:

```
function V=mulq(A,B)
V = [ A(1)*B(1) - A(2)*B(2) - A(3)*B(3) - A(4)*B(4), ...
      A(1)*B(2) + A(2)*B(1) + A(3)*B(4) - A(4)*B(3), ...
      A(1)*B(3) + A(3)*B(1) + A(4)*B(2) - A(2)*B(4), ...
      A(1)*B(4) + A(4)*B(1) + A(2)*B(3) - A(3)*B(2) ];
end

function V=invq(A)
AA = A(1)*A(1) + A(2)*A(2) + A(3)*A(3) + A(4)*A(4);
V = [ A(1), -A(2), -A(3), -A(4) ]/AA;
end

function V=conq(A)
V = [ A(1), -A(2), -A(3), -A(4) ];
end
```

Along with, `mulq(A,B)` and `invq(A)`, we've also included a convenient function for taking the conjugate: `conq(A)`.

```
>> global a b c;
>> A = [ 2, 1, 0, -1 ]; % a = 2 + 1i + 0j - 1k
>> B = [ -3, 0, 1, 1 ]; % b = -3 + 0i + 1j + 1k
>> C = [ 1, 0, -1, 1 ]; % c = 1 + 0i - 1j + 1k
>>
>> % ...using ver.1 code to solve #2 quadratic example...
>> % ...transforming on 'initialization'...
>> % call psolqabc(A,-ABA^-1,ACA^-1)
>> psolqabc(A,-mulq(A,mulq(B,inv(A))),mulq(A,mulq(C,inv(A))));
>> % ...solve as before, then get solutions...
>> Q1 = r1*psolqv(r1) - mulq(B,inv(A)) % add displacement to old soln, Q1 = q1 - BA^-1
>> Q2 = r2*psolqv(r2) - mulq(B,inv(A)) % add displacement to old soln, Q2 = q2 - BA^-1
>>
```

By substition and working out, we could also observe that,

$$q(A, -ABA^{-1}, ACA^{-1}) = -A \cdot q(A, B, C) \cdot A^{-1} \tag{2.40}$$

$$\therefore \quad Q(A, B, C) = -Aq(A, B, C)A^{-1} - BA^{-1} = -(Aq(A, B, C) + B)A^{-1} \tag{2.41}$$

So, rather than transforming the input parameters on 'initalization,' we can make all the transformations on 'completion,' after the intermediate solution is found, working things out the following way instead;

```
>>
>> % ...using ver.1 code to solve #2 quadratic example...
>> % ...transforming on 'completion'...
>> % call psolqabc(A,B,C)
>> psolqabc(A,B,C);
>> % ...solve as before, then get solutions...
>> Q1 = -mulq(mulq(A,r1*psolqv(r1)) + B,invq(A)) % using, Q1 = -(A.q1 + B)/A
>> Q2 = -mulq(mulq(A,r2*psolqv(r2)) + B,invq(A)) % using, Q2 = -(A.q2 + B)/A
>>
```

The only thing to remember when using these alternatives, is that we don't get the magnitude, λ, for the two solutions right away. We have to compute them, $\lambda_1 = |Q_1|$, and, $\lambda_2 = |Q_2|$; since, the magnitudes computed, r_1 and r_2, are now for the intermediate solution, and not the final solution magnitudes we need. Also, if we want to compare plots of the curves, just using the 'ver.2' code will show the correct graph without any further adjustments. But, using `mulq(A,B)` and `invq(A)` and these transform variable methods, provide for a quick check, and verification on the algorithms and their implementations, when writing code. It always helps to be able to do things in a different way, and to compare results.

```
>> % verify results...
>> V1 = mulq(Q1,mulq(Q1,A)) + mulq(Q1,B) + C % verify, Q1.Q1.A + Q1.B + C = 0
>> V2 = mulq(Q2,mulq(Q2,A)) + mulq(Q2,B) + C % verify, Q2.Q2.A + Q2.B + C = 0
>>
```

Looked at this way, the new solution can be seen as the result of a composite transformation, made up of a 'rotation', AqA^{-1}, followed by an 'inversion', $(-1)\cdot AqA^{-1}$, followed by a 'translation', $Q = (-1)\cdot AqA^{-1} - BA^{-1}$; or, translation then inversion, $Q = (-1)\cdot(AqA^{-1} + BA^{-1})$, for the final two steps.

Let us now look at another type of equation transformation, the exchange symmetry, in the two given parameters, a and c.

`Exchange Symmetry` (a, c)`:`

Start with the first quadratic and it's unit quaternion solution.

$$aq^2 + bq + c = 0 \tag{2.42}$$

$$q = \frac{(|a|^2a^* - c^*ac^*)(cb^*a - ba^*a) + (|c|^2c^* - a^*ca^*)(cb^*c - ba^*c)}{(|a|^2 - |c|^2)((|a|^2 + |c|^2)^2 - ((ac^*) + (ac^*)^*)^2)} \tag{2.43}$$

Multiply on the right by the square of the inverse, $1/q^2 = (q^*)^2$. We get a new quadratic, where the two parameters, a and c, have exchanged roles. Now, c is the lead, and a is the constant. So, if, $q = q(a, b, c)$, is the formula of solution for first equation, then we have, $q^* = q(c, b, a)$, for the solution to the conjugate. We just plug in "c" in the slot for "a," and plug in "a" in the slot for "c," and we can write down immediately the solution to this new quadratic.

$$a + bq^* + c(q^*)^2 = 0 \tag{2.44}$$

$$q^* = \frac{(|c|^2c^* - a^*ca^*)(ab^*c - bc^*c) + (|a|^2a^* - c^*ac^*)(ab^*a - bc^*a)}{(|c|^2 - |a|^2)((|c|^2 + |a|^2)^2 - ((ca^*) + (ca^*)^*)^2)} \tag{2.45}$$

Since this is the conjugate of our original, q, we can just get back a formula for the, q, by now conjugating the (a, c)–swapped solution. So, let's see how that works out.

$$q = (q^*)^* = \left(\frac{(|c|^2c^* - a^*ca^*)(ab^*c - bc^*c) + (|a|^2a^* - c^*ac^*)(ab^*a - bc^*a)}{(|c|^2 - |a|^2)((|c|^2 + |a|^2)^2 - ((ca^*) + (ca^*)^*)^2)}\right)^* \tag{2.46}$$

$$\tag{2.47}$$

$$q = \frac{(ab^*c - bc^*c)^*(|c|^2c^* - a^*ca^*)^* + (ab^*a - bc^*a)^*(|a|^2a^* - c^*ac^*)^*}{(|c|^2 - |a|^2)((|c|^2 + |a|^2)^2 - ((ca^*) + (ca^*)^*)^2)} \tag{2.48}$$

$$\tag{2.49}$$

$$q = \frac{(c^*ba^* - c^*cb^*)(|c|^2c - ac^*a) + (a^*ba^* - a^*cb^*)(|a|^2a - ca^*c)}{(|c|^2 - |a|^2)((|c|^2 + |a|^2)^2 - ((ca^*) + (ca^*)^*)^2)} \tag{2.50}$$

$$\tag{2.51}$$

$$q = \frac{(a^*cb^* - a^*ba^*)(|a|^2a - ca^*c) + (c^*cb^* - c^*ba^*)(|c|^2c - ac^*a)}{(|a|^2 - |c|^2)((|a|^2 + |c|^2)^2 - ((ac^*) + (ac^*)^*)^2)} \tag{2.52}$$

The denominator is just a scalar. Conjugation doesn't affect it. But, we can optionally rearrange the composite scalar, $S(ca^*) = (ca^* + (ca^*)^*)/2$, in a number of alternate forms, $S(ca^*) = S(a^*c) = S(ac^*) = S(c^*a)$, so, let's pick the form that corresponds best with our original formula, and compare.

$$q = \frac{(|a|^2a^* - c^*ac^*)(cb^* - ba^*)a + (|c|^2c^* - a^*ca^*)(cb^* - ba^*)c}{(|a|^2 - |c|^2)((|a|^2 + |c|^2)^2 - ((ac^*) + (ac^*)^*)^2)} \qquad \text{"original"} \tag{2.53}$$

$$q = \frac{a^*(cb^* - ba^*)(|a|^2a - ca^*c) + c^*(cb^* - ba^*)(|c|^2c - ac^*a)}{(|a|^2 - |c|^2)((|a|^2 + |c|^2)^2 - ((ac^*) + (ac^*)^*)^2)} \qquad \text{"new"} \tag{2.54}$$

Now the conjugate of any real number is just that real number itself, $\forall \mu \in \mathbb{R},\ \mu^* = \mu$, in particular, $0^* = 0$, so we can conjugate the entire equation to find,

$$(a + bq^* + c(q^*)^2 = 0)^* \quad \equiv \quad (a^* + qb^* + q^2c^* = 0) \tag{2.55}$$

Hence when we swap, a and c, and then replace, a, b, c, by their conjugates, a^*, b^*, c^*, we have,

$$q^2a + qb + c = 0 \tag{2.56}$$

$$q = \frac{(|c|^2c - ac^*a)(a^*bc^* - b^*cc^*) + (|a|^2a - ca^*c)(a^*ba^* - b^*ca^*)}{(|c|^2 - |a|^2)((|c|^2 + |a|^2)^2 - ((c^*a) + (c^*a)^*)^2)} \qquad \text{“\#1-soln original-subs”} \tag{2.57}$$

$$q = \frac{c(a^*b^* - bc)(|c|^2c^* - a^*ca^*) + a(a^*b^* - bc)(|a|^2a^* - c^*ac^*)}{(|c|^2 - |a|^2)((|c|^2 + |a|^2)^2 - ((c^*a) + (c^*a)^*)^2)} \qquad \text{“\#1-soln new-subs”} \tag{2.58}$$

$$q = \frac{(|a|^2a - ca^*c)(b^*ca^* - a^*ba^*) + (|c|^2c - ac^*a)(b^*cc^* - a^*bc^*)}{(|a|^2 - |c|^2)((|a|^2 + |c|^2)^2 - (ac^* + (ac^*)^*)^2)} \qquad \text{“\#2-soln”} \tag{2.14}$$

which yield yet another formula expressions for the solutions to this reverse quadratic equation. This means we have four distinct ways to arrive at the solution to the second (#2) reverse quadratic, $q^2a + qb + c = 0$. We can either: (I) solve directly using the two-hand algebra; or use the solution to the first quadratic (#1), $aq^2 + bq + c = 0$, in one of three ways: (II) transform on initialization, (III) transform on completion, (IV) apply the swap exchange operation $(a, c) \to (c, a)$ followed by parameter conjugation $(a, b, c) \to (a^*, b^*, c^*)$. For the purpose of this paper, we shall continue with the solution from method (I), in the following analysis and testing.

TESTING 2

the example....here....(working papge 01 – to be completed)

#2-Reference: – the complete solution.

$$q^2a + qb + c = 0 \qquad a, b, c, q \in \mathbb{H}_R \tag{2.59}$$

$$m_1, m_2, m_3 \in \mathbb{R},\ m_1^2 + m_2^2 + m_3^2 = 1 \tag{2.60}$$

$b = 0:$ (2.61)

$q^2a + c = 0$ (2.62)

Case 1: $ca^* \in \mathbb{R},\ ca^* > 0.:$ $\quad q = \sqrt{|c|/|a|} \cdot (m_1 i + m_2 j + m_3 k)$ (2.63)

Case 2: $ca^* \in \mathbb{R},\ ca^* < 0.:$ $\quad q = \pm\sqrt{|c|/|a|} \cdot 1$ (2.64)

Case 3: $ca^* \notin \mathbb{R},\ S(ca^*) = 0.:$ $\quad q = \pm\sqrt{|c|/|a|} \cdot (1 - ca^*/|ca^*|)/\sqrt{2}$ (2.65)

Case 4: $ca^* \notin \mathbb{R},\ S(ca^*) \neq 0.:$ $\quad q = \pm\sqrt{|c|/|a|} \cdot (1 - ca^*/|ca^*|/)/\sqrt{2 - (ca^* + (ca^*)^*)/|ca^*|}$ (2.66)

$b \neq 0:$ (2.67)

$$p = p(\lambda) = \frac{\lambda^5(\lambda^4 aa^*a - ca^*c)(b^*c - \lambda^2 a^*b)a^* + \lambda(cc^*c - \lambda^4 ac^*a)(b^*c - \lambda^2 a^*b)c^*}{(\lambda^4|a|^2 - |c|^2) \cdot |\lambda^2 a - c|^2 \cdot |\lambda^2 a + c|^2} \tag{2.68}$$

$|p(\lambda)| = 1$ (2.69)

$$q = \lambda \cdot p = \lambda \cdot \frac{\lambda^5(\lambda^4 aa^*a - ca^*c)(b^*c - \lambda^2 a^*b)a^* + \lambda(cc^*c - \lambda^4 ac^*a)(b^*c - \lambda^2 a^*b)c^*}{(\lambda^4|a|^2 - |c|^2) \cdot |\lambda^2 a - c|^2 \cdot |\lambda^2 a + c|^2} \tag{2.70}$$

Onplane Magic Factors: $|p(\lambda)| < 1,\ \ \lambda = \sqrt{|c|/|a|},\ \ a, b, c \in \mathbb{C} \subset \mathbb{H}_R$ (2.71)

$$q_1 \ = \ \lambda \cdot u_1 \ = \ \lambda \cdot p \cdot \left(1 + \frac{(p - p^*)}{|p - p^*|}\sqrt{|p|^{-2} - 1}\right) \tag{2.72}$$

(2.73)

$$q_2 \ = \ \lambda \cdot u_2 \ = \ \lambda \cdot p \cdot \left(1 - \frac{(p - p^*)}{|p - p^*|}\sqrt{|p|^{-2} - 1}\right)$$

Offplane Magic Factors: $|p(\lambda)| < 1,\ \ \lambda = \sqrt{|c|/|a|},\ \ a, b, c \in \mathbb{H}_R$ (2.74)

$$q_1 = \lambda \cdot u_1 = \lambda \cdot \left(p + u \cdot \sqrt{\frac{|(\lambda p)^2 a + (\lambda p)b + c|}{|c|}}\right) \tag{2.75}$$

$$q_2 = \lambda \cdot u_2 = \lambda \cdot \left(p - u \cdot \sqrt{\frac{|(\lambda p)^2 a + (\lambda p)b + c|}{|c|}}\right) \tag{2.76}$$

$$w = \lambda^2((\lambda p)^2 a + (\lambda p)b + c)a^* \ , \qquad \lambda = \sqrt{\frac{|c|}{|a|}} \tag{2.77}$$

where, $u = u(w)$: (2.78)

Case 1: $w \in \mathbb{R},\ w > 0.:$ $\quad u = (m_1 i + m_2 j + m_3 k)$ (2.79)

Case 2: $w \in \mathbb{R},\ w < 0.:$ $\quad u = 1$ (2.80)

Case 3: $w \notin \mathbb{R},\ S(w) = 0.:$ $\quad u = (1 - w/|w|)/\sqrt{2}$ (2.81)

Case 4: $w \notin \mathbb{R},\ S(w) \neq 0.:$ $\quad u = (1 - w/|w|/)/\sqrt{2 - (w + w^*)/|w|}$ (2.82)

#3. A third quadratic equation.

Consider the following quadratic equation ($a, b, c, q, \in \mathbb{H}_R$),

$$qaq + bq + c = 0 \ , \qquad b \neq 0 \tag{3.1}$$

As it is, in the non-abelian world, things that have just one form in abelian algebra, turn out to have multiple forms when variables non-commute. So, here we have yet another simple quadratic equation, expressing the same theme of the sum of the square, the linear, and the constant, so familiar in abelian algebra. This time, the leading parameter, a, is sandwiched between the two $q's$ forming the square. This leads to yet another solution. So, let's explore this to compare. We start out the same way as before, converting to a pair of linear equations in, q and q^*.

$$qa + cq^* = -b \tag{3.2}$$
$$qc^* + a^*q^* = -b^* \tag{3.3}$$

Changing over to $\mathbb{X}$-*algebra*, this becomes,

$$a'\hat{q} + c\hat{q}^* = -\hat{b} \tag{3.4}$$
$$c^{*\prime}\hat{q} + a^*\hat{q}^* = -\hat{b}^* \tag{3.5}$$

By now, we know the drill. Eliminate, $\hat{q}^*$! But, things are a little different this time. There's a new twist. Both, c and a^*, that pre-multiply the conjugate, $\hat{q}^*$, are of the same hand, $c, a^* \in \mathbb{H}_R$. They don't commute. So, we can't just flip them around, like we did in previous examples. This time, we need an extra multiplication step. So, first we multiply the top equation, by c^*, to ensure a scalar pre-multiplies the $\hat{q}^*$. Then, we multiply the top equation again, this time, by a^*, to line up with the lower equation. Finally, we multiply the bottom equation by the scalar, $|c|^2 = c^*c$, to match terms with the top. Then, we're ready to eliminate.

$$a^*c^*a'\hat{q} + a^*c^*c\hat{q}^* = -a^*c^*\hat{b} \tag{3.6}$$
$$|c|^2c^{*\prime}\hat{q} + |c|^2a^*\hat{q}^* = -|c|^2\hat{b}^* \tag{3.7}$$

Subtracting the lower from the upper,

$$(a^*c^*a' - |c|^2c^{*\prime})\hat{q} = -a^*c^*\hat{b} + |c|^2\hat{b}^* \tag{3.8}$$

So, now, our hexpe factor, $h = (a^*c^*a' - |c|^2c^{*\prime})$, has 2-terms with triple products, whereas, last time we dealt with only pair products. Nevertheless, things work pretty much the same way. We proceed to calculate the right-conjugate factor, $h^{*R} = (a^*c^*a' - |c|^2c^{*\prime})^{*R}$, and multiply both sides by this first,

$$(a^*c^*a' - |c|^2c^{*\prime})^{*R} \cdot (a^*c^*a' - |c|^2c^{*\prime})\hat{q} = (a^*c^*a' - |c|^2c^{*\prime})^{*R} \cdot (-a^*c^*\hat{b} + |c|^2\hat{b}^*) \tag{3.9}$$
$$(caa' - |c|^2c^{*\prime}) \cdot (a^*c^*a' - |c|^2c^{*\prime})\hat{q} = (caa' - |c|^2c^{*\prime}) \cdot (-a^*c^*\hat{b} + |c|^2\hat{b}^*) \tag{3.10}$$

expanding the left side factor, and slipping out the extra, $|c|^2$, to one side; which now becomes possible,

$$|c|^2 \cdot (|a|^2(a')^2 - caa'c^{*\prime} - a^*c^*c^{*\prime}a' + |c|^2(c^{*\prime})^2)\hat{q} = (caa' - |c|^2c^{*\prime}) \cdot (-a^*c^*\hat{b} + |c|^2\hat{b}^*) \tag{3.11}$$

Continuing, we construct the left-conjugate factor, $(|a|^2(a')^2 - caa'c^{*\prime} - a^*c^*c^{*\prime}a' + |c|^2(c^{*\prime})^2)^{*L}$, and multiply both sides of the equation with it,

$$|(|a|^2(a^{*\prime})^2 - cac'a^{*\prime} - a^*c^*a^{*\prime}c' + |c|^2(c')^2) \cdot |c|^2 \cdot (|a|^2(a')^2 - caa'c^{*\prime} - a^*c^*c^{*\prime}a' + |c|^2(c^{*\prime})^2)\hat{q} \tag{3.12}$$
$$= (|a|^2(a^{*\prime})^2 - cac'a^{*\prime} - a^*c^*a^{*\prime}c' + |c|^2(c')^2) \cdot (caa' - |c|^2c^{*\prime}) \cdot (-a^*c^*\hat{b} + |c|^2\hat{b}^*) \tag{3.13}$$

multiplying out, and rearranging,

$$q = \left(\begin{array}{c} |a|^2 \cdot c \cdot (a^*aaa^{*\prime} - c^*a^{*\prime}a^{*\prime}c^{*\prime} - acac' - c^*a^{*\prime}c'a') \\ \hline +|c|^2 \cdot c \cdot (ac'a^{*\prime}c^{*\prime} + c^*a^*c^*a^{*\prime} + ac'c'a' - cc^*c^*c') \\ \hline |c|^2 \cdot (|a|^2 - |c|^2)^2 \cdot ((|a|^2 + |c|^2)^2 - ((ca) + (ca)^*)^2) \end{array} \right) \cdot (-a^*c^*\hat{b} + |c|^2\hat{b}^*) \tag{3.14}$$

and apart from a few extra factors, of $|c|^2$, c^*, and c, which we had to put in to get the otherwise non-abelian factors to move around in the right way in the expressions, formulas workout similar to before. Completing,

$$q \;=\; \frac{(cab^* - aa^*b)(|a|^2a^* - cac) + (c^*cb^* - a^*c^*b)(|c|^2c - a^*c^*a^*)}{(|a|^2 - |c|^2)\cdot((|a|^2 + |c|^2)^2 - ((ca) + (ca)^*)^2)} \tag{3.15}$$

This is the '*unit magnitude*' solution, for the special case where the parameters, a, b, c, allow such unit quaterion, q, with, $|q| = 1$, for a solution. Again, we rescale this solution, to obtain the more general solution. Setting, $q = \lambda p$, with, $p, q \in \mathbb{H}_R,\ |p| = 1,\ \lambda \in \mathbb{R},\ \lambda > 0$, we obtain the transformed quadratic equation,

$$p\lambda^2 ap + \lambda bp + c = 0 \tag{3.16}$$

which, because it has similar form, has the same solution, this time in the unknown direction unit, p, with new known parameters obtained by the replacements, $a \to \lambda^2 a,\ b \to \lambda b,\ c \to c$. When substituted into the given solution, this gives, immediately, the solution for, p, viz.,

$$p \;=\; \frac{(\lambda^3 cab^* - \lambda^5 aa^*b)(\lambda^6 a^*aa^* - \lambda^2 cac) + (\lambda c^*cb^* - \lambda^3 a^*c^*b)(cc^*c - \lambda^4 a^*c^*a^*)}{(\lambda^4|a|^2 - |c|^2)\cdot((\lambda^4|a|^2 + |c|^2)^2 - \lambda^4((ca) + (ca)^*)^2)} \tag{3.17}$$

We now consider p to be a function of λ, $p = p(\lambda)$, and subject it to the constraint requirement, $|p(\lambda)| = 1$, or, equivalently, $p(\lambda)p(\lambda)^* = 1$; find the values of λ that satisfy this constraint, and then use each such λ found to write a solution to the original quadratic equation,

$$q \;=\; \lambda \cdot \frac{(\lambda^3 cab^* - \lambda^5 aa^*b)(\lambda^6 a^*aa^* - \lambda^2 cac) + (\lambda c^*cb^* - \lambda^3 a^*c^*b)(cc^*c - \lambda^4 a^*c^*a^*)}{(\lambda^4|a|^2 - |c|^2)\cdot((\lambda^4|a|^2 + |c|^2)^2 - \lambda^4((ca) + (ca)^*)^2)} \tag{3.18}$$

$$\;=\; Tq \cdot Uq \tag{3.19}$$

The denominator is once again our familiar expression, with a slightly different conjugation profile in the composite scalar $[S(ca) = ((ca) + (ca)^*)/2]$; ca instead of ac^*. There is flexibility in writing composite scalars like this, because of the cyclical scalar permuatation rule[8], and the laws of conjugation, so that, $S(ac^*) = S(c^*a) = S(a^*c) = S(ca^*)$. But the alternative collection of comparable expressions, $S(ca) = S(ac) = S(a^*c^*) = S(c^*a^*)$, are in a different set.

Example in Quaternions (#3)

In preparation for implementation in code, let us first rearrange numerator and denominator in the unit direction,

$$p \;=\; \frac{\lambda^5(cab^* - \lambda^2 aa^*b)(\lambda^4 a^*aa^* - cac) + \lambda(c^*cb^* - \lambda^2 a^*c^*b)(cc^*c - \lambda^4 a^*c^*a^*)}{(\lambda^4|a|^2 - |c|^2)\cdot|\lambda^2 a^* - c|^2\cdot|\lambda^2 a^* + c|^2} \tag{3.20}$$

In our previous denominator, the lower factor with composite scalar factored into the form, $|\lambda^2 a - c|^2 \cdot |\lambda^2 a + c|^2$, which zeros out at exactly the same critical point, $\lambda = \sqrt{|c|/|a|}$, whenever the, "c and a," are proportional to each other, and when they are not, neither of these subfactors vanish for any value of λ. So, they contributed nothing 'extra' to our results there. Here again, in this third example, we see this same composite expression contributes nothing additional to the already determined critical point. This time, the expression factors into, $|\lambda^2 a^* - c|^2 \cdot |\lambda^2 a^* + c|^2$, and zeros out again at the same place, $\lambda = \sqrt{|c|/|a|}$. But now, this happens whenever the, "c and a^*," are proportional, and when they are not, neither subfactor vanishes for any value of λ. Here then, is the replacement code for 'ver.3' of the two MATLAB functions, `psolqd(r)` and `psolqev(Q)`.

---

[8] $S(abcd) = S(dabc) = S(cdab) = S(bcda)$.

in MATLAB code:

```
function p=psolqd(r)     % ver.3, (qaq + bq + c = 0)
global a b c;
np = r^5*(c*a*b' - r^2*a*a'*b)*(r^4*a'*a*a' - c*a*c);
np = np + r*(c'*c*b' - r^2*a'*c'*b)*(c*c'*c - r^4*a'*c'*a');
dp = trace((r^4*a*a' - c*c')*(r^2*a' - c)*(r^2*a' - c)'*(r^2*a' + c)*(r^2*a' + c)')/2;
p = np/dp;
end

function V=psolqev(Q)     % ver.3, (qaq + bq + c = 0)
global a b c;
q = [(Q(1,1)+1i*Q(1,2)),(Q(1,3)+1i*Q(1,4));-(Q(1,3)+1i*Q(1,4))',(Q(1,1)+1i*Q(1,2))'];
v = q*a*q + b*q + c;
V = [real(v(1,1)) , imag(v(1,1)), real(v(1,2)), imag(v(1,2))];
end
```

discussion of equation transformations...

TESTING 3

the example here.....(working paper 01 — to be completed)

`#3-Reference`: – the complete solution. ???

$$\underline{qaq + bq + c = 0} \qquad a, b, c, q \in \mathbb{H}_R \tag{3.21}$$

(3.22)

(working paper 01 – to be completed)

#4. A fourth quadratic equation.

Consider the following quadratic equation ($a, b, c, q, \in \mathbb{H}_R$),

$$qaq + qb + c = 0 \ , \qquad b \neq 0 \tag{4.1}$$

(working paper 01 – to be completed)

#5. A fifth quadratic equation.

Consider the following quadratic equation ($a, b, c, q, \in \mathbb{H}_R$),

$$aq^2 + qb + c = 0 \quad , \qquad b \neq 0 \tag{5.1}$$

This is referred to as a two-sided quadratic, and can also be written in the generic form, $q^2 + aqb + c = 0$, with unit coefficient on the square term. We have known coefficients, a, b, c, appearing both on the left side and on the right side of the unknown parameter, e.g. aq^2 and qb, or aqb, and it turns out that, because of this, we can't simplify the 4-D equation to some 1-D form, and obtain the usual, $|p(\lambda)| = 1$, that would enable us to find the whole solution conveniently from the determination of a single variable, λ, by just finding where a 1-D curve cuts a straight line, as done before.

(working paper 01 – to be completed)

#6. A sixth quadratic equation.

Consider the following quadratic equation ($a, b, c, q, \in \mathbb{H}_R$),

$$q^2 a + bq + c = 0 \quad , \qquad b \neq 0 \tag{6.1}$$

This is another two-sided quadratic equation, which also can be written in the alternate generic form, $q^2 + aqb + c = 0$, with unit coefficient on the square term; it thus can be written also in the previous form, $aq^2 + qb + c = 0$.

Of the six simplist equations, then, four are completely solvable, by reduction to the 1-D problem, $|p(\lambda)| = 1$, and have the solutions as given in the sections #1-to-#4 above, for all values of the known parameters, a, b, c. While the last two are unsolvable with this method.

$$\#1: \; aq^2 + bq + c = 0 \qquad \checkmark \; \forall a, b, c \in \mathbb{H}, \qquad |p(\lambda)| = 1 \tag{6.2}$$
$$\#2: \; q^2 a + qb + c = 0 \qquad \checkmark \; \forall a, b, c \in \mathbb{H}, \qquad |p(\lambda)| = 1 \tag{6.3}$$
$$\#3: \; qaq + bq + c = 0 \qquad \checkmark \; \forall a, b, c \in \mathbb{H}, \qquad |p(\lambda)| = 1 \tag{6.4}$$
$$\#4: \; qaq + qb + c = 0 \qquad \checkmark \; \forall a, b, c \in \mathbb{H}, \qquad |p(\lambda)| = 1 \tag{6.5}$$
$$\#5: \; aq^2 + qb + c = 0 \qquad ??? \tag{6.6}$$
$$\#6: \; q^2 a + bq + c = 0 \qquad ??? \tag{6.7}$$

The check ✓ indicates we can determine whether or not there's a solution, and where there's a solution we have found the formulas.

Now, the most general three term quadratic, with one square term, one linear term, and a constant, can always be re-expressed as one of these six simple equations. The more general equation is sometimes expressible in one of the first four forms, which always have solutions via reduction to a 1-D problem, but sometimes the general equation can only be transformed into one of the last two forms, which are currently unsolvable.

; e.g., we can convert to the #5 equation:

$$a_1 q a_2 q a_3 + b_1 q b_2 + c = 0 \tag{6.8}$$
$$\downarrow$$
$$a_1 q a_2 q + b_1 q b_2 {a_3}^{-1} + c {a_3}^{-1} = 0 \tag{6.9}$$
$$\text{Let,} \quad Q = a_2 q \tag{6.10}$$
$$\therefore \quad q = {a_2}^{-1} Q \tag{6.11}$$
$$a_1 {a_2}^{-1} Q a_2 {a_2}^{-1} Q + b_1 {a_2}^{-1} Q b_2 {a_3}^{-1} + c {a_3}^{-1} = 0 \tag{6.12}$$
$$a_1 {a_2}^{-1} Q^2 + b_1 {a_2}^{-1} Q b_2 {a_3}^{-1} + c {a_3}^{-1} = 0 \tag{6.13}$$
$$a_2 {b_1}^{-1} a_1 {a_2}^{-1} Q^2 + Q b_2 {a_3}^{-1} + a_2 {b_1}^{-1} c {a_3}^{-1} = 0 \tag{6.14}$$
$$\downarrow$$
$$aq^2 + qb + c = 0 \tag{6.15}$$

So, if we can solve #5, then we can solve all. This form, $aq^2 + qb + c = 0$, can then be considered the key quadratic equation for the general three-term, i.e. the simplified canonical form, that represents the entire set of possible three-term quadratics; or perhaps we'd like to use the alternate form, $q^2 + aqb + c = 0$, for this purpose.

#7. A more general quadratic equation.

GENERALIZING.

The most general type of quadratic equation in the skew-field of right-hand quaternions takes the form,

$$\sum_{j=1}^{n} a_{j1} \cdot q \cdot a_{j2} \cdot q \cdot a_{j3} \;+\; \sum_{k=1}^{m} b_{k1} \cdot q \cdot b_{k2} \;+\; c \;=\; 0 \qquad a_{ji}, b_{ji}, c, q \in \mathbb{H}_R \tag{7.1}$$

Solutions to all of these quatratics can be found by generalizing the methods of this paper. The criterion for solvability via reduction to a 1-D problem, is that the quaternion unknown, q, be fully exposed on the right, or fully exposed on the left, or that the equation can be converted to such, by a suitable transformation. By solvability, of course, we mean by using the methods outlined here. (It is always possible to break out the four components of the quaternion parameters, and re-express the problem as a set of four simultaneous quadratic equations in four real valued unknowns, (t, x, y, z), and use the methods of multi-variable calculus to arrive at solutions.) Thus, the equation must be capable of being put into one of the two following forms;

$$\sum_{j=1}^{n} a_{j1} \cdot q \cdot a_{j2} \cdot q \;+\; \sum_{k=1}^{m} b_{k1} \cdot q \;+\; c \;=\; 0 \tag{7.2}$$

$$\sum_{j=1}^{n} q \cdot a_{j2} \cdot q \cdot a_{j3} \;+\; \sum_{k=1}^{m} q \cdot b_{k2} \;+\; c \;=\; 0 \tag{7.3}$$

In this case, we can simply divide on the right, or divide on the left, by the unit, q, to obtain the linear equation in the direction unit. Otherwise, we're unable to transform the quatratic equation into a pair of linear equations, in q and q^*. Here, there's no advantage in including the distinct b_{ki}-parameters, since, by the summation, they can be trivially combined into a single known quaternion, b. Therefore, the most general equation of interest to us, for the moment, will take one of the forms,

$$\sum_{j=1}^{n} a_{j1} \cdot q \cdot a_{j2} \cdot q \;+\; b \cdot q \;+\; c \;=\; 0 \qquad ,\; b \neq 0 \tag{7.4}$$

$$\sum_{j=1}^{n} q \cdot a_{j2} \cdot q \cdot a_{j3} \;+\; q \cdot b \;+\; c \;=\; 0 \qquad ,\; b \neq 0 \tag{7.5}$$

Consider the first of these equations. We transform to linear system. (Now that we have only one type of summation, we can leave out the limits to improve readability. From here on, the indexed sums are all: $j = 1, \cdots, n$.)

$$\begin{aligned} \sum a_{j1} \cdot q \cdot a_{j2} \;+\; c \cdot q^* \qquad\qquad &= \; -b \\ q \cdot c^* \;+\; \sum a^*_{j2} \cdot q^* \cdot a^*_{j1} \qquad &= \; -b^* \end{aligned} \tag{7.6}$$

Converting from $\mathbb{H}_R$-*algebra* to $\mathbb{X}$-*algebra* this becomes,

$$\begin{aligned} \left(\sum a_{j1} \cdot a'_{j2}\right) \cdot \hat{q} \;+\; c \cdot \hat{q}^* \qquad &= \; -\hat{b} \\ c^{*\prime} \cdot \hat{q} \;+\; \left(\sum a^*_{j2} \cdot {a^*_{j1}}'\right) \cdot \hat{q}^* \qquad &= \; -\hat{b}^* \end{aligned} \tag{7.7}$$

There are now general hexpe numbers in our expressions. We have, $c \in \mathbb{H}_R$, and, $c^{*\prime} \in \mathbb{H}_L$, as before, in previous solution steps. But, now we have two new parameters in, $\mathbb{X}_n$, owing to our generalization of the quadratic equation to include n terms of square form, $aqaq$, where previously we dealt with only one such term. However, despite this increase in complexity, the solution method is essentially the same as before. The only major difference now, is that we must use the advanced method of inverting the hexpe number, which we've called *the Gilgamesh Solution*. We also need a couple of extra multiplications steps in the expression manipulations. We must first multiply the top equation by the conjugate, c^*, to ensure we have a scalar pre-multiplying the term, $\hat{q}^*$, and not a right handed quaternion, $c \in \mathbb{H}_R$, which would otherwize get stuck there. This scalar will then commute with all numbers in the next step, which is to multiply the top equation again, this time by the n-term sum, $\sum a^*_{j2} \cdot {a^*_{j1}}'$, from the lower equation, with

its mix of right and left parameters. The bottom equation is just multiplied on the left by the scalar, $|c|^2 = c^* c$, to match terms with the top. We then obtain,

$$\begin{array}{rcrcl} \left(\sum a^*_{j2} \cdot a^{*\prime}_{j1}\right) \cdot c^* \cdot \left(\sum a_{j1} \cdot a'_{j2}\right) \cdot \hat{q} & + & \left(\sum a^*_{j2} \cdot a^{*\prime}_{j1}\right) \cdot c^* c \cdot \hat{q}^* & = & -\left(\sum a^*_{j2} \cdot a^{*\prime}_{j1}\right) \cdot c^* \cdot \hat{b} \\ |c|^2 c^{*\prime} \cdot \hat{q} & + & |c|^2 \left(\sum a^*_{j2} \cdot a^{*\prime}_{j1}\right) \cdot \hat{q}^* & = & -|c|^2 \hat{b}^* \end{array} \tag{7.8}$$

Subtracting the lower from upper, changing one j index to k, to keep indices distinct, we get,

$$\left(\left(\sum a^*_{j2} \cdot a^{*\prime}_{j1}\right) \cdot c^* \cdot \left(\sum a_{k1} \cdot a'_{k2}\right) - |c|^2 c^{*\prime} \right) \cdot \hat{q} = \left(|c|^2 \hat{b}^* - \left(\sum a^*_{j2} \cdot a^{*\prime}_{j1}\right) \cdot c^* \cdot \hat{b} \right) \tag{7.9}$$

There are now '$n \times n + 1$' terms in the hexpe coefficient, $h \in \mathbb{X}_{n \times n+1}$, which explains why we need the advanced inversion method here.

$$h \cdot \hat{q} = (|c|^2 \hat{b}^* - \left(\sum a^*_{j2} c^* \cdot a^{*\prime}_{j1}\right) \cdot \hat{b}) \tag{7.10}$$

$$\tag{7.11}$$

$$h = -|c|^2 c^{*\prime} + \sum a^*_{j2} \cdot a^{*\prime}_{j1} \cdot c^* \cdot a_{k1} \cdot a'_{k2} \tag{7.12}$$

$$= -|c|^2 c^{*\prime} + \sum a^*_{j2} c^* a_{k1} \cdot a^{*\prime}_{j1} a'_{k2} \tag{7.13}$$

$$\tag{7.14}$$

Now, the two indices, j, k, both range over n. We should point out that, although it seems formidably complex, with such a great number as $(n \times n + 1)$ terms in the expression for, h, things are not so difficult as might first seem. As discussed in APPENDIX B, any bilinear expression formed of right-hand times left-hand product terms, may be converted into a similar summation with just 4 terms or less: $h \in \mathbb{X}_{n \times n+1} \to h \in \mathbb{X}_4$. From a practical point of view, then, we never have to deal with the apparent complexity when solving numerically. Once we know the numerical values of the coefficients, $\{a_{jk}, c\}$, we can simply combine them, just as we combined the, b_{ki}, coefficients into a single, b. Only here, we'd combine into a 4-term factor like, $h = \alpha_1 \beta'_1 + \alpha_2 \beta'_2 + \alpha_3 \beta'_3 + \alpha_4 \beta'_4$. This does require us to know the values of the coefficients. But, that's usually the case when we come to actually solve real problems. At times, however, it is useful to carry through the computations with the original symbolic parameters, especially, when seeking to understand the structure of the solution, and to help with interpretation and attributing meaning.

Solving the equation, now requires computation of the inverse, h^{-1}, which usually can no longer be by the simple conjugated cube we used before, $(h^{*R} h)^{*L} h^{*R}$, instead we use the advanced *Gilgamesh Solution*, to write $\hat{q}$, as;

$$\hat{q} = h^{-1} \cdot (|c|^2 \hat{b}^* - \left(\sum a^*_{j2} c^* \cdot a^{*\prime}_{j1}\right) \cdot \hat{b}) \tag{7.15}$$

$$= \left[\frac{h^*(hh^*)^{*R} + 2(h^{*L}(h^{*R} h^{*L})^{*R})^{*R}}{h^*(hh^*)^{*R} h + 2(h^{*L}(h^{*R} h^{*L})^{*R})^{*R} h} \right] \cdot (|c|^2 \hat{b}^* - \left(\sum a^*_{j2} c^* \cdot a^{*\prime}_{j1}\right) \cdot \hat{b}) \tag{7.16}$$

This is the *unit magnitude solution*, that assumes that the coefficients, $\{a_{ji}, c, b\}$, are such that the unit quaternion solution, $|q| = 1$, exists. It gives us the solution to the direction. Again, as before, we re-scale the unknown parameter, setting, $q = \lambda p$, with, $|p| = 1, p \in \mathbb{H}_R$, and, the magnitude, $\lambda \in \mathbb{R}$, with, $\lambda > 0$. Our quadratic equation becomes,

$$\sum_{j=1}^{n} a_{j1} \cdot (\lambda p) \cdot a_{j2} \cdot (\lambda p) + b \cdot (\lambda p) + c = 0 \tag{7.17}$$

$$\sum_{j=1}^{n} (\lambda a_{j1}) \cdot p \cdot (\lambda a_{j2}) \cdot p + (\lambda b) \cdot p + c = 0 \tag{7.18}$$

And we now have an equation in the unit quaterion, p, with any arbitrary values for, $\{a_{ji}, c, b\}$, so, by making the replacements, $a_{ji} \to \lambda a_{ji}, b \to \lambda b, c \to c$, we can immediately write down the general solution to the direction,

$$\hat{p} = h(\lambda)^{-1} \cdot (\lambda |c|^2 \hat{b}^* - \lambda^3 \left(\sum a^*_{j2} c^* \cdot a^{*\prime}_{j1}\right) \cdot \hat{b}) \tag{7.19}$$

$$\tag{7.20}$$

Resolving this, moving the $\mathbb{H}_L$ specific parameters over to the r.h.s of the, $\hat{b}$ and $\hat{b}^*$, parameters, removing the $\hat{}$ hat marks, enables us to express the unit magnitude in $\mathbb{H}_R$ and to write the constraint equation that determines the magnitude, λ, as,

$$p(\lambda)p(\lambda)^* = 1 \tag{7.21}$$

Then, solving for, λ, that satisfies this constraint, we write the solution to the complete quaternion, for each of the unique values of λ found, as,

$$q = \lambda \cdot p(\lambda) \tag{7.22}$$

$$\begin{array}{lcl} h & = & -|c|^2 \cdot c^{*\prime} + \sum a^*_{j,2} c^* a_{k,1} \cdot a^*_{j,1}{}' a'_{k,2} \\ h^{*R} & = & -|c|^2 \cdot c^{*\prime} + \sum a^*_{k,1} c\, a_{j,2} \cdot a^*_{j,1}{}' a'_{k,2} \\ h^{*L} & = & -|c|^2 \cdot c' \ + \sum a^*_{j,2} c^* a_{k,1} \cdot a'_{k,2}{}^* a'_{j,1} \\ h^* = (h^{*R})^{*L} & = & -|c|^2 \cdot c' \ + \sum a^*_{k,1} c\, a_{j,2} \cdot a'_{k,2}{}^* a'_{j,1} \end{array} \tag{7.23}$$

$$hh^* = +|c|^6 - |c|^2 \cdot \sum a^*_{k,1} c\, a_{j,2} \cdot c^{*\prime} a'_{k,2}{}^* a'_{j,1} - |c|^2 \cdot \sum a^*_{j,2} c^* a_{k,1} \cdot a^*_{j,1}{}' a'_{k,2} c' \tag{7.24}$$

$$+ \sum a^*_{p,2} c^* a_{q,1} a^*_{k,1} c\, a_{j,2} \cdot a^*_{p,1}{}' a'_{q,2} a'_{k,2}{}^* a'_{j,1} \tag{7.25}$$

EXAMPLE - **A Symbolic Example**

Let us now do an example of a specific problem to see how things work out in the symbolic computations. We choose the simplest equation, that's more advanced than those we treated before, and so pick a quadratic equation with two square terms. The challenge then, is to solve the following equation.

$$a_{11} \cdot \boldsymbol{q} \cdot a_{12} \cdot \boldsymbol{q} \ + \ a_{21} \cdot \boldsymbol{q} \cdot a_{22} \cdot \boldsymbol{q} \ + \ b \cdot \boldsymbol{q} \ + \ c \ = \ 0 \tag{7.26}$$

$$\tag{7.27}$$

In the first step, we presume a unit quaternion solution exists and so convert the quadratic equation to a system of linear equations in two variables, $\boldsymbol{q}, \boldsymbol{q}^*$.

$$\begin{array}{rcl} a_{11} \cdot \boldsymbol{q} \cdot a_{12} + a_{21} \cdot \boldsymbol{q} \cdot a_{22} & + & c \cdot \boldsymbol{q}^* = -b \\ \boldsymbol{q} \cdot c^* & + & a^*_{12} \cdot \boldsymbol{q}^* \cdot a^*_{11} + a^*_{22} \cdot \boldsymbol{q}^* \cdot a^*_{21} = -b^* \end{array} \tag{7.28}$$

In the second step, we convert from Hamilton's 19th century $\mathbb{H}_R$-*algebra* to our new 21st century $\mathbb{X}$-*algebra*,

$$\begin{array}{rcl} (a_{11} \cdot a'_{12} + a_{21} \cdot a'_{22}) \cdot \hat{\boldsymbol{q}} & + & c \cdot \hat{\boldsymbol{q}}^* = -\hat{b} \\ c^{*\prime} \cdot \hat{\boldsymbol{q}} & + & (a^*_{12} \cdot a^*_{11}{}' + a^*_{22} \cdot a^*_{21}{}') \cdot \hat{\boldsymbol{q}}^* = -\hat{b}^* \end{array} \tag{7.29}$$

In the third step, we multiply the top and bottom equations by suitable factors to prepare to eliminate the unknown conjugate, $\hat{\boldsymbol{q}}^*$,

$$\begin{array}{rcl} (a^*_{12}a^*_{11}{}' + a^*_{22}a^*_{21}{}')c^*(a_{11}a'_{12} + a_{21}a'_{22}) \cdot \hat{\boldsymbol{q}} & + & (a^*_{12}a^*_{11}{}' + a^*_{22}a^*_{21}{}')|c|^2 \cdot \hat{\boldsymbol{q}}^* = -(a^*_{12}a^*_{11}{}' + a^*_{22}a^*_{21}{}')c^*\hat{b} \\ |c|^2 c^{*\prime} \cdot \hat{\boldsymbol{q}} & + & |c|^2(a^*_{12}a^*_{11}{}' + a^*_{22}a^*_{21}{}') \cdot \hat{\boldsymbol{q}}^* = -|c|^2\hat{b}^* \end{array} \tag{7.30}$$

In the fourth step, we subtract the bottom equation from the top, to obtain an expression for just the, $\hat{\boldsymbol{q}}$,

$$(\ (a^*_{12}a^*_{11}{}' + a^*_{22}a^*_{21}{}')c^*(a_{11}a'_{12} + a_{21}a'_{22}) - |c|^2 c^{*\prime}\ ) \cdot \hat{\boldsymbol{q}} = (\ |c|^2\hat{b}^* - (a^*_{12}a^*_{11}{}' + a^*_{22}a^*_{21}{}')c^*\hat{b}\ ) \tag{7.31}$$

In the fifth step, we define, $h \in \mathbb{X}_5$, and compute the various right and left conjugates of, h, it's squares and cubes, as needed for the procedure of inversion to find, h^{-1}.

$$\begin{array}{rcl} h \cdot \hat{\boldsymbol{q}} & = & (\ |c|^2\hat{b}^* - (a^*_{12}a^*_{11}{}' + a^*_{22}a^*_{21}{}')c^*\hat{b}\ ) \\ h & = & (\ (a^*_{12}a^*_{11}{}' + a^*_{22}a^*_{21}{}')c^*(a_{11}a'_{12} + a_{21}a'_{22}) - |c|^2 c^{*\prime}\ ) \\ & = & (-|c|^2 c^{*\prime} + a^*_{12}c^*a_{11} \cdot a^*_{11}{}'a'_{12} + a^*_{12}c^*a_{21} \cdot a^*_{11}{}'a'_{22} + a^*_{22}c^*a_{11} \cdot a^*_{21}{}'a'_{12} + a^*_{22}c^*a_{21} \cdot a^*_{21}{}'a'_{22}) \end{array} \tag{7.32}$$

$$\begin{array}{rcl} h^{*R} & = & (-|c|^2 c^{*\prime} + a^*_{11}ca_{12} \cdot a^*_{11}{}'a'_{12} + a^*_{21}ca_{12} \cdot a^*_{11}{}'a'_{22} + a^*_{11}ca_{22} \cdot a^*_{21}{}'a'_{12} + a^*_{21}ca_{22} \cdot a^*_{21}{}'a'_{22}) \\ h^{*L} & = & (-|c|^2 c' + a^*_{12}c^*a_{11} \cdot a^*_{12}{}'a'_{11} + a^*_{12}c^*a_{21} \cdot a^*_{22}{}'a'_{11} + a^*_{22}c^*a_{11} \cdot a^*_{12}{}'a'_{21} + a^*_{22}c^*a_{21} \cdot a^*_{22}{}'a'_{21}) \\ h^* = (h^{*R})^{*L} & = & (-|c|^2 c' + a^*_{11}ca_{12} \cdot a^*_{12}{}'a'_{11} + a^*_{21}ca_{12} \cdot a^*_{22}{}'a'_{11} + a^*_{11}ca_{22} \cdot a^*_{12}{}'a'_{21} + a^*_{21}ca_{22} \cdot a^*_{22}{}'a'_{21}) \end{array}$$

$$hh^* = \tag{7.33}$$

$$\begin{aligned}
&+|c|^6 - |c|^2.a_{12}^* c^* a_{11} \cdot a_{11}^{*\prime} a_{12}^{\prime} c^{\prime} - |c|^2.a_{12}^* c^* a_{21} \cdot a_{11}^{*\prime} a_{22}^{\prime} c^{\prime} - |c|^2.a_{22}^* c^* a_{11} \cdot a_{21}^{*\prime} a_{12}^{\prime} c^{\prime} - |c|^2.a_{22}^* c^* a_{21} \cdot a_{21}^{*\prime} a_{22}^{\prime} c^{\prime} \\
&-|c|^2.a_{11}^* c a_{12} \cdot c^{*\prime} a_{12}^{*\prime} a_{11}^{\prime} + a_{12}^* c^* a_{11} a_{11}^* c a_{12} \cdot a_{11}^{*\prime} a_{12}^{\prime} a_{12}^{*\prime} a_{11}^{\prime} + a_{12}^* c^* a_{21} a_{11}^* c a_{12} \cdot a_{11}^{*\prime} a_{22}^{\prime} a_{12}^{*\prime} a_{11}^{\prime} \\
&\qquad + a_{22}^* c^* a_{11} a_{11}^* c a_{12} \cdot a_{21}^{*\prime} a_{12}^{\prime} a_{12}^{*\prime} a_{11}^{\prime} + a_{22}^* c^* a_{21} a_{11}^* c a_{12} \cdot a_{21}^{*\prime} a_{22}^{\prime} a_{12}^{*\prime} a_{11}^{\prime} \\
&-|c|^2.a_{21}^* c a_{12} \cdot c^{*\prime} a_{22}^{*\prime} a_{11}^{\prime} + a_{12}^* c^* a_{11} a_{21}^* c a_{12} \cdot a_{11}^{*\prime} a_{12}^{\prime} a_{22}^{*\prime} a_{11}^{\prime} + a_{12}^* c^* a_{21} a_{21}^* c a_{12} \cdot a_{11}^{*\prime} a_{22}^{\prime} a_{22}^{*\prime} a_{11}^{\prime} \\
&\qquad + a_{22}^* c^* a_{11} a_{21}^* c a_{12} \cdot a_{21}^{*\prime} a_{12}^{\prime} a_{22}^{*\prime} a_{11}^{\prime} + a_{22}^* c^* a_{21} a_{21}^* c a_{12} \cdot a_{21}^{*\prime} a_{22}^{\prime} a_{22}^{*\prime} a_{11}^{\prime} \\
&-|c|^2.a_{11}^* c a_{22} \cdot c^{*\prime} a_{12}^{*\prime} a_{21}^{\prime} + a_{12}^* c^* a_{11} a_{11}^* c a_{22} \cdot a_{11}^{*\prime} a_{12}^{\prime} a_{12}^{*\prime} a_{21}^{\prime} + a_{12}^* c^* a_{21} a_{11}^* c a_{22} \cdot a_{11}^{*\prime} a_{22}^{\prime} a_{12}^{*\prime} a_{21}^{\prime} \\
&\qquad + a_{22}^* c^* a_{11} a_{11}^* c a_{22} \cdot a_{21}^{*\prime} a_{12}^{\prime} a_{12}^{*\prime} a_{21}^{\prime} + a_{22}^* c^* a_{21} a_{11}^* c a_{22} \cdot a_{21}^{*\prime} a_{22}^{\prime} a_{12}^{*\prime} a_{21}^{\prime} \\
&-|c|^2.a_{21}^* c a_{22} \cdot c^{*\prime} a_{22}^{*\prime} a_{21}^{\prime} + a_{12}^* c^* a_{11} a_{21}^* c a_{22} \cdot a_{11}^{*\prime} a_{12}^{\prime} a_{22}^{*\prime} a_{21}^{\prime} + a_{12}^* c^* a_{21} a_{21}^* c a_{22} \cdot a_{11}^{*\prime} a_{22}^{\prime} a_{22}^{*\prime} a_{21}^{\prime} \\
&\qquad + a_{22}^* c^* a_{11} a_{21}^* c a_{22} \cdot a_{21}^{*\prime} a_{12}^{\prime} a_{22}^{*\prime} a_{21}^{\prime} + a_{22}^* c^* a_{21} a_{21}^* c a_{22} \cdot a_{21}^{*\prime} a_{22}^{\prime} a_{22}^{*\prime} a_{21}^{\prime}
\end{aligned} \tag{7.34}$$

$$\tag{7.35}$$

$$\tag{7.36}$$

$$(hh^*)^{*R} = \tag{7.37}$$

$$h^*(hh^*)^{*R} = \tag{7.38}$$

$$h^*(hh^*)^{*R}h = \tag{7.39}$$

$$h^{*R}h^{*L} = \tag{7.40}$$

$$(h^{*R}h^{*L})^{*R} = \tag{7.41}$$

$$h^{*L}(h^{*R}h^{*L})^{*R} = \tag{7.42}$$

$$(h^{*L}(h^{*R}h^{*L})^{*R})^{*R} = \tag{7.43}$$

$$(h^{*L}(h^{*R}h^{*L})^{*R})^{*R}h = \tag{7.44}$$

$$\tag{7.45}$$

(working paper 01 – to be completed)

EXAMPLE - **A Numeric Example**

(working paper 01 – to be completed)

8. CUBIC EQUATIONS.

Consider the following cubic equation in quaternions ($a, b, c, d, q \in \mathbb{H}_R$),

$$aq^3 + bq^2 + cq + d = 0 \tag{8.1}$$

It is natural to ask, how far can we take this "*two-hand two-step*" method? Can we use our knowledge about solving the linear, and now the quadratic equation, to find solutions, similarly, for the cubic, and perhaps move up the chain to nth degree polynomials? After all, it worked that way before in abelian algebra. Over the past centuries, the mysteries of the whole system of polynomials were gradually unravelled, finding that closed form solutions exist up to the 4th degree, and producing the *Fundamental theorem of algebra*, which tells us that the nth degree polynomial has exactly n roots, counting multiplicity, of course. Here, however, in the particular non-abelian algebra of quaternions, we've already seen quadratic equations with an infinity of roots! So, already, huge differences abound. But, there are also quaternion equations with a finite number of roots. There *is* order and structure amidst the chaos. So, this becomes a matter of classifying the equations into types, based on their form, parameter range values, and the ability to transform reversibly or irreversibly into other equations.

Let's then, take a look at this simple cubic. We try our trick, and multiply on the right by, $1/q$, considering the four known parameters, a, b, c, d, are such that unit quaternion solution exists, so, $1/q = q^*$, with, $|q| = 1$, and we obtain the following pair of quadratic and conjugate equations,

$$aq^2 + bq + c + dq^* = 0 \tag{8.2}$$
$$(q^*)^2a^* + q^*b^* + c^* + qd^* = 0 \tag{8.3}$$

Well, we can't have both, q^2 and $(q^*)^2$, square terms, AND both, q and q^*, linear terms, in just one pair of simultaneous equations. That doesn't work well for us. If we're treating, q and q^*, as a pair of unknowns, seeking to eliminate one, with "linear algebra methods," then we'd better get rid of one of the square terms. Hence, we now multiply the bottom equation on the left by, q, to remove the square, $(q^*)^2$, conveniently using the fact that, $qq^* = 1$, here, so things work out just fine.

$$q(q^*)^2a^* + qq^*b^* + qc^* + qqd^* = 0 \tag{8.4}$$
$$q^*a^* + b^* + qc^* + q^2d^* = 0 \tag{8.5}$$

So, now we have just the three, q^2, q, q^*, parameters to deal with. Let's rearrange the final pair of simultaneous equations to see what we've got a so far;

$$aq^2 + bq + dq^* = -c \tag{8.6}$$
$$q^2d^* + qc^* + q^*a^* = -b^* \tag{8.7}$$

We can now eliminate the conjugate, q^*, by multiplying the top equation on the right by, a^*, and the bottom equation on the left by, d, then subtracting.

$$aq^2a^* + bqa^* + dq^*a^* = -ca^* \tag{8.8}$$
$$dq^2d^* + dqc^* + dq^*a^* = -db^* \tag{8.9}$$
$$\tag{8.10}$$
$$aq^2a^* - dq^2d^* + bqa^* - dqc^* + ca^* - db^* = 0 \tag{8.11}$$

Success! We're able to transform our cubic equation in one variable, of single cubic term, into a quadratic equation in one variable, with multiple quadratic terms, for the special case where the variables, a, b, c, d, happen to take on just the right values to allow a unit quaternion solution, q, with, $|q| = 1$, to exist. So, maybe we can find *some* roots? Can we solve all polynomial equations, with our new 21st century $\mathbb{X}$-*algebra* methods? These things will be explored in a future paper, as they are out of the scope of this one.

9. CONCLUSIONS.

The square root of a number, like, $\sqrt{2}$, can generally only be computed by applying an algorithm. One could use, for example, the well known *bisection algorithm*, to find this root of 2. Thus, even though we tend to think of the formula, $z = (-b \pm \sqrt{b^2 - 4ac})/(2a)$, as the quintessential "closed form" expression of solution to the polynomial, in this case of degree 2, i.e., $az^2 + bz + c = 0,\ z \in \mathbb{C}$, and seek to find such "closed form" expressions for other polynomials of higher degree, it still often requires an elementary algorithm to compute the actual *numerical* result, in the end. Moving on up to quaternions, the corresponding quadratic equation, $aq^2 + bq + c = 0,\ q \in \mathbb{H}$, also has a "closed form" solution.

$$q \quad = \quad \lambda \cdot \frac{(\lambda^6|a|^2a^* - \lambda^2c^*ac^*)(\lambda^3cb^*a - \lambda^5ba^*a) + (|c|^2c^* - \lambda^4a^*ca^*)(\lambda cb^*c - \lambda^3ba^*c)}{(\lambda^4|a|^2 - |c|^2)((\lambda^4|a|^2 + |c|^2)^2 - \lambda^4((ac^*) + (ac^*)^*)^2)} \tag{9.1}$$

Only that, now, we need to know the magnitude, λ, to plug into this formula, to find the complete quaternion solution. Finding that λ, to complete the solution, usually requires an algorithm. But, the significant thing here to note, is that the same *bisection algorithm* that will determine the $\sqrt{2}$, e.g., $\texttt{Solve[}f(\lambda) = \lambda^2/2 - 1 = 0\texttt{]}$, to complete the abelian polynomial solutions, is also all we need to apply to the non-abelian polynomials to determine the complete solution, i.e. $\texttt{Solve[}f(\lambda) = p(\lambda)p(\lambda)^* - 1 = 0\texttt{]}$, where, $p(\lambda) = q/\lambda$. Thus, although looking a bit more formidable than the formulas in commuting algebra, these newer non-commuting formulas to the quadratic require *no higher order of computational complexity*,[9] than the older commuting ones that came before. We can now state, with *almost* complete certainty, that all quadratic equations, at least, of the three forms,

$$ah^2 + bh + c = 0, \quad h^2a + hb + c = 0, \quad hah + bh + c = 0,$$

whether over reals, $h = x \in \mathbb{R}$, complex numbers, $h = z \in \mathbb{C}$, or quaternions, $h = q \in \mathbb{H}$, have solutions that can be found by a *native* "closed form" formula plus a simple "bisection algorithm." By the reference "*native*," we mean that the hypercomplex number does not have to be broken out into its individual sub-components to express the closed form result! The *native* numbers appear *undivided* in the expressions.[10]

(working paper 01 – to be completed)

Acknowledgements
The idea for the solution method presented in this paper came to me after reading Tian's paper ([YT1]), where the equation, $ax - x^*b = c$, was discussed. After solving the first two of Tian's examples, using the new two-hand method, in order to illustrate the new method in my last paper ([PJ4]), it occurred to me, sometime later, while thinking about quadratics, that the very equation Tian discussed with x and x^* could be used to tackle quadratic equations by simply restricting the x and x^* to unit quaternions, and then rescaling the magnitudes to find the general solutions. So, thanks to Tian, for setting up the light bulb, making this particular equation prominant by publishing a discussion on it. That got me thinking in the right direction.

AUTHOR's NOTE: I've been sitting on this text for many years, being unable to find the time to get back to it, to complete the work. So, I decided to remove lots of unfinished parts, and create a much smaller text with just the most complete sections as "working paper 01," to release the ideas to anyone who might find this interesting enough to follow up on them. – pmj

---

[9] Here *order of complexity* refers to the number, n, of iterations required to obtain the mth decimal place of numeric accuracy.
[10] Write, $|q| = \lambda$, then the non-abelian closed form formula, $q = q(|q|; a, b, c)$, is *a partly implicit function*, compared to the abelian completely *explicit function.*, $z = z(a, b, c)$. Only the unit direction function, $p(\lambda; a, b, c) = q/\lambda$, is truely closed form in *explicit* sense.

APPENDIX A: Reference Calculations

Multimedia Appendix to the .pdf document. A Video of Calculations.
The contents of this appendix are not in the text of this paper, but rather included in an accompanying 'silent' video in .mp4 format.[11]
The video has the title:

`Solving Quaternion Quadratic Equations, A Reference Calculations Video.`

Reference Calculations

Objectives:
1. To exhibit in step-by-step fashion the methods of non-abelian manipulation.
2. To provide a more detailed record of the calculations used to obtain the results of this paper.
3. To make this paper accessible to a wider audience than just the specialists in the field.

The Reference Calculation Video contains the step-by-step derivations for all equations from the start of a section, up to the "boxed" equation of that section. Arriving at the expressions in the boxes is the most challenging part of the work, partly because of the novel nature of the method being used, and partly because of the nature of non-abelian calculations in general. The rest of the paper contains familiar mathematics, and the reader is expected to have no trouble with the content following on from these boxed equations.

Also included in the text of this paper, in APPENDIX D below, is a method of derivation using the right hand algebra alone, for the three quadratic equations, $aq^2 + bq + c = 0$, $q^2a + qb + c = 0$, $qaq + bq + c = 0$, as illustrations in the alternate art of reckoning within Hamilton's original calculus. While the *discovery* of such solutions is easier using two hands, it is sometimes possible to arrive at the same results with just one hand alone, as these three exhibits there illustrate. However, it is currently more challenging to start out in one hand algebra, and find the solution, especially when it is still unknown what the solution should look like. Once one has seen the form of solution, however, finding the alternate path to that solution using one hand algebra is much less challenging.

(working paper 01 – this multimedia "video" was planned, but not yet produced...!) .end

[11] Although the video is 'voice silent,' it does have an audio track. The most appropriate background audio turned out to be the syncopated keyboard sounds. The recorded sounds are the uniquely characteristic snychronized 'clay tile' clacking of the Fujitsu Happy Hacking Professional 2 Keyboard, as the author pounds away on the keys, typing in the expressions appearing on the screen.

APPENDIX B: Notation Conventions !

From $\mathbb{H}$-algebra to $\mathbb{X}$-algebra.

The following represents the notation we use in our papers when going from $\mathbb{H}$-algebra to $\mathbb{X}$-algebra and back, in the process of solving quaternion problems. It is to be understood that the letters, i, j, k, generally refer to the quaternion basis units of Hamilton's system, and are normally anti-commuting in right-handed fashion, with his 1843 rules: $i = jk = -kj, j = ki = -ik, k = ij = -ji$, so that, $ijk = -1$. Occasionally, we use the triple, i, j, k, to refer to the general solution to $\sqrt{-1}$ where the hand is *ambiguous* and can either be left or right handed, so we'll often indicate, $ijk = -1$, or, $ijk = +1$, to refer to the *handedness* established for the basis when needed. Apart from specifying the sign on the unit of the ijk triple, the *anti-commuting rules*, and other things like *associativity*, *distibutivity*, and the fact that the units have *negative squares*, the idea of *conjugation*, with its $*$*-algebra* rules, and so on, are all assumed to be known and are not usually referenced in our definitions.

Here then are the definitions of the symbols we use:

$$\begin{aligned}
&\mathbb{H}_R \equiv \mathbb{H} = \{q \mid q = w1 + xi + yj + zk; w, x, y, z \in \mathbb{R}, ijk = -1\}\\
&\mathbb{H}_L \equiv \mathbb{H}' = \{q \mid q = w1 + xi' + yj' + zk'; w, x, y, z \in \mathbb{R}, i'j'k' = +1\}\\
&\mathbb{X}_n = \{h \mid h = a_1 b'_1 + a_2 b'_2 \cdots a_n b'_n, a_j \in \mathbb{H}_R, b'_j \in \mathbb{H}_L\},\ \ n = 1, 2, 3, \cdots\\
&\mathbb{X}_b^+ = \{1, i, j, k, i', j', k', ii', jj', kk', jk', ki', ij', kj', ik', ji' \ ;\ i, j, k \in \mathbb{H}_R, i', j', k' \in \mathbb{H}_L\}\\
&\mathbb{X}_b^- = \mathbb{X}_b - \mathbb{X}_b^+\\
&\mathbb{X}_b = \{\pm 1, \pm i, \pm j, \pm k, \pm i', \pm j', \pm k', \pm ii', \pm jj', \pm kk', \pm jk', \pm ki', \pm ij', \pm kj', \pm ik', \pm ji'; i, j, k \in \mathbb{H}_R, i', j', k' \in \mathbb{H}_L\}\\
&\mathbb{X} = \{h \mid h = \sum_j a_j e_j \ ;\ e_j \in \mathbb{X}_b^+, a_j \in \mathbb{R}\}
\end{aligned} \tag{b-1}$$

The standard notation, $\mathbb{H}$, for Hamilton's quaternions, is augmented by subscripts to emphasize that the distinctions between right-hand, $\mathbb{H}_R$, and left-hand, $\mathbb{H}_L$, play a significant role in the methods we develop and present. The usual view that right and left are isomorphic algebras, often gives the impression that these two are equivalent, thus only one need be considered, leading to the prior neglect of the left-hand.

Note that, $\mathbb{X}_n \subseteq \mathbb{X}$. For, $n < 4$, $\mathbb{X}_n \subset \mathbb{X}$, while, for, $n >= 4$, $\mathbb{X}_n = \mathbb{X}$. It is not always possible to write an arbitrary element, $h \in \mathbb{X}$, of our $\mathbb{X}$-*algebra*, in bilinear form, $a_1 b'_1 + \cdots + a_n b'_n$, with small number of terms, n. But, the method of solution we can apply to problems depends on our ability to represent the h factor as a specific sum of such terms. When, $n = 1, 2$, we can invert, $h \to h^{-1}$, using the single conjugated cube, $(h^{*R}h)^{*L}h^{*R}$, but for larger n we have to use the more complicated *Gilgamesh Solution* formula which combines conjugated cubes. Given that there are 16 elements in the basis, $\mathbb{X}_b^+$, we can always re-write, $h \in \mathbb{X}_{16}$, since all the units are trivally already in the bilinear format, $a_j b'_j$. So, it is immediately obvious we can always reduce the n to no more than 16. Although, at times it may not be convenient to do so. However, these are unit elements, and by combining them into whole quaternions we can do even better. The following example shows one way that the 16 term bilinear expression could be further reduced to just 4 terms. Thus, every 4×4 matrix over $\mathbb{R}$ can be refactored into the form of $h \in \mathbb{X}_4$.

$$\begin{aligned}
h &= h_0 1 + h_{1R}\boldsymbol{i} + h_{2R}\boldsymbol{j} + h_{3R}\boldsymbol{k} + h_{1L}\boldsymbol{i'} + h_{2L}\boldsymbol{j'} + h_{3L}\boldsymbol{k'} + h_{1M}\boldsymbol{ii'} + h_{2M}\boldsymbol{jj'} + h_{3M}\boldsymbol{kk'}\\
&\quad + h_{1A}\boldsymbol{jk'} + h_{2A}\boldsymbol{ki'} + h_{3A}\boldsymbol{ij'} + h_{1Z}\boldsymbol{kj'} + h_{2Z}\boldsymbol{ik'} + h_{3Z}\boldsymbol{ji'}\\
&= (h_0 1 + h_{1R}\boldsymbol{i} + h_{2R}\boldsymbol{j} + h_{3R}\boldsymbol{k}) \cdot 1 + (h_{1L} + h_{1M}\boldsymbol{i} + h_{3Z}\boldsymbol{j} + h_{2A}\boldsymbol{k}) \cdot \boldsymbol{i'}\\
&\quad + (h_{2L} + h_{3A}\boldsymbol{i} + h_{2M}\boldsymbol{j} + h_{1Z}\boldsymbol{k}) \cdot \boldsymbol{j'} + (h_{3L} + h_{2Z}\boldsymbol{i} + h_{1A}\boldsymbol{j} + h_{3M}\boldsymbol{k}) \cdot \boldsymbol{k'}\\
&= a_1 b'_1 + a_2 b'_2 + a_3 b'_3 + a_4 b'_4 \qquad\qquad a_j \in \mathbb{H}_R, b'_j \in \mathbb{H}_L
\end{aligned} \tag{b-2}$$

The R-L-M-A-Z designations in the subscript labels refer to the RIGHT-LEFT-META-ALPHA-ZETA naming convention, introduced in our previous papers, for the five distinct ijk basis triplets into which the basis is partitioned. This then extends our ijk notation to unit numbers beyond the quaternions, but we generally clarify this by adding subscripts. Thus, we also write the above, $h \in \mathbb{X}$, with following single letter notation,

$$\begin{aligned}
h &= h_0\mathbf{1} + h_{1R}\boldsymbol{i_R} + h_{2R}\boldsymbol{j_R} + h_{3R}\boldsymbol{k_R} + h_{1L}\boldsymbol{i_L} + h_{2L}\boldsymbol{j_L} + h_{3L}\boldsymbol{k_L} + h_{1M}\boldsymbol{i_M} + h_{2M}\boldsymbol{j_M} + h_{3M}\boldsymbol{k_M}\\
&\quad + h_{1A}\boldsymbol{i_A} + h_{2A}\boldsymbol{j_A} + h_{3A}\boldsymbol{k_A} + h_{1Z}\boldsymbol{i_Z} + h_{2Z}\boldsymbol{j_Z} + h_{3Z}\boldsymbol{k_Z}
\end{aligned} \tag{b-3}$$

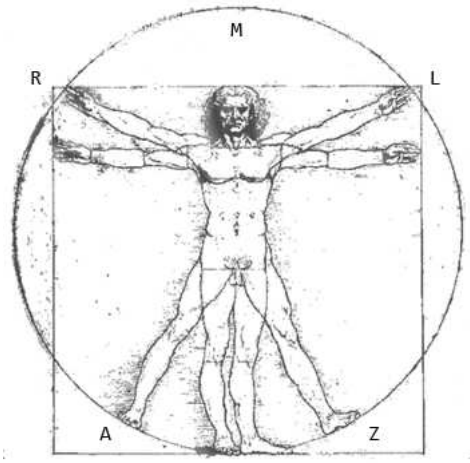


FIG. 13: Leonardo da Vinci's The Vitruvian Man

`Anthropomorphic Numbers.` Our "*man in the matrix*" designations for R-L-M-A-Z help to remember the hypercomplex units, their properties and functions. The five algebras are like the five limbs of the human body:
i.e. $\mathbf{R} \equiv$ `right hand`, $\mathbf{L} \equiv$ `left hand`, $\mathbf{M} \equiv$ `head`, $\mathbf{A} \equiv$ `right foot`, $\mathbf{Z} \equiv$ `left foot`.
They share common functions. R and L are like the right-hand and left-hand. Just as an olympic ice skater demonstrates the art of rotating the body by lowering and raising her right and left arms, to increase and decrease her spin rate, so too do the R and L quaternions act as the generators of rotation in geometric operations. And just as the man may stoop down and couch into a small form, or stand up and stretch out enlarging the apparence of his form, so too do the A and Z numbers generate scaling operations of shrinking and expanding in the geometry. So, although the ALPHA and ZETA numbers are somewhat like right and left algebras, they do not generate rotations like RIGHT and LEFT quaternions, they perform scaling operations instead, which is more similar to the right-foot and left-foot function of the human form. The META numbers are interesting, as they too produce scaling operations, but they posess an axial symmetry, having neither right nor left bias, much like the central position of the human head. One can amusingly think that the brain *would* expand and contract at will, were it not for the fact that nature produced such a hard skeletal shell to contain the brain, confining man's thoughts within a fixed skull. That doesn't stop the mind from expanding and contracting, however, as men are often capable of thinking on more lofty issues at times, or reflecting on rather narrow concerns at others. Of course, one could view the human form to be, not just this Leonardo da Vinci's Pentagram of The Vitruvian Man, but the result of two opposite interlocking pentagrams, one largely visible in matter the other mostly invsible in spirit, and reflect on other amusing correspondences of function and form.

When contemplating these correspondences, one should include consideration of the interesting result that manifests in this hypercomplex system when applying the $\mathbb{X}$-*algebra* to represent operations in affine geometry. As shown in a previous paper ([PJ2]), the operation of *translation* involves the four 'limbs' R-L-A-Z, but not the head, M. One can think of this translation operation as a man walking down the road pushing his right foot and left foot and swinging his right hand and left hand for balance to enable his movement. All four limbs are involved when walking down a path, even though we tend to think of walking requiring only the two feet. The inclusion of the two hands obviously reduces the energy required to move, and increases efficency in the translation operation; a feature that has caught the attention of researchers studying motion dynamics in the relatively new field of *Underactuated Robotics*. It is noteworthy, then, that the algebra requires all four limbs to participate to effect the translation in affine geometry.

So, apart from being a memory aid, one could take this anthrophormiphsm nore seriously. One can imagine biological systems emerging out of the void, needing to adapt to the 4-d structure, selectively incorporating the features required to function, matching the geometric order encountered in nature. To move around in the 4-dimensional spacetime world, the lifeform adopted the most efficient expression of form to enable movement. Since movement is essentially made up of sequences of transformations of form within the 4-d spacetime, the order inherent in the transformations, as represented by the symmetry operations that make up transformations, provided a natural pathway for expressions in evolutionary adaptation. Thus the lifeform evolved limbs to provide functions correlating to the order in the geometric algebra. The geometric symmetries in transformations, of course, are represented algebraically by that which we call a mathematical group. The quaternions form one such group, and are natural tools to construct representations of geometric transformations.

`Symmetry Groups.` In a 4-dimensional spacetime world, there are 4 degrees of freedom to move about. If we erect 4 principal axes, E, I, J, K, to measure things, there are then 8 natural units required, $\{\pm E, \pm I, \pm J, \pm K\}$, and the maximal group structure possible here then consists of 8 elements. These "*Groups of Order 8*," as it is referred to in the literature, can only be formed in 5 distinct flavors. If they all share the same unit element, E, then there are just 5 distinct triples, IJK, possible to describe the movements of form through spacetime.

GROUP ELEMENT COMPARISON

1..Q	$\equiv$	$[\mathbf{1}, \boldsymbol{i_R}, \boldsymbol{j_R}, \boldsymbol{k_R}]$	−4D, SPACETIME, quaternion group
2..D4	$\equiv$	$[\mathbf{1}, \boldsymbol{i_A}, \boldsymbol{i_R}, \boldsymbol{k_M}]$	−2D, SQUARE, dihedral group
3..Z2 × Z2 × Z2	$\equiv$	$[\mathbf{1}, \boldsymbol{i_M}, \boldsymbol{j_M}, \boldsymbol{k_M}]$	−3D, CUBOID
4..Z2 × Z4	$\equiv$	$[\mathbf{1}, \boldsymbol{i_R}, \boldsymbol{i_L}, \boldsymbol{i_M}]$	−3D, SQUARE CUBOID subgroup
5..Z8	$\equiv$	$[\mathbf{1}, \boldsymbol{i_R}, (\mathbf{1}+\boldsymbol{i_R})/\sqrt{2}, (\mathbf{1}-\boldsymbol{i_R})/\sqrt{2}]$	−2D, OCTAGON subgroup, cyclic group 8

(b-4)

* The 5 Groups of Order 8 and their relatiohship to the 5 R-L-M-A-Z triples. *

It is possible to point to some familiar objects in the phenomenological world whose *geometric symmetries* are reflected in the *algebraic properties* of the mathematical groups constructed from the R-L-M-A-Z basis elements. One question of interest is how many dimensions are required before we can find an object that contains the geometric symmetry characteristic of a particular group. We see that the five groups require two 2D geometries, two 3D, and one 4D. The quaternions are true 4D symmetries, there being no lower dimensional geometric objects that exhibit quaternion group structure either as group or subgroup of its collection of symmetry operations.

The R quaternions are 4D, and the L quaternions are 4D. When we bring two 4D objects together they intersect in a space of one lower dimension, i.e. 3D, hence, it is not surprising that we find symmetries of 3D objects when we combine the left hand and right hand units, to form the other M, A, and Z units. When we combine three 4D spaces together they intersect in a space of two lower, i.e. 2D, and again, we find symmetries of 2D objects among the basis. Depending on how we pick our triplet of units, IJK, then, from among the plain or composite basis unit alternatives, we find 4D, or 3D, or 2D, indications, emerging as a result.

Of course, breaking a matrix down into its possible symmetry groups, has the advantage of identifying the components of the related transformation by invariant characteristics. We think of a rigid object, which is translating and rotating simultaneously, as possessing two types of motion, each involving an unique symmetry, and each following an independent conservation law related to that symmetry. Translation has conservation of linear momentum, while rotation has conservation of angular momentum. Partitioning the matrix into an hypercomplex number then, is much like fourier series expansion of a function, that breaks down the function in terms of simpler oscillating sines and cosines, that has the power to reveal which of the simpler elements in the composition are more or less significant to the overall transformation. So, whether we expand a function by fourier series, or use some other orthogonal basis functions, or expand a transformation matrix by the group elements of some hypercomplex number basis, we are generally seeking a convenient description of the phenomena at hand, to aid insight into the processes at work.

The complete basis, $\mathbb{X}_b$, forms a non-abelian group of order 32. It directly contains 4 of the 5 groups of order 8, as indicated above, and the 5th group is an elementary linear combination of the basis elements. The elements of $\mathbb{X}_b$ all have orders, 1, 2, or 4, but not 8. The cyclic group Z8 needs that order 8 element. But, it can be constructed from the linear combination of an order 1 and an order 4 element, as shown above.

`The Distinguished Left-Hand.` Special mention should be made of the fact that the left-hand units, i', j', k', are not just any left hand basis, they are *linked* to the chosen right-hand basis, i, j, k, in a particular way. There is a special distinguished left-handed basis, that corresponds to a given right-handed basis, different from all the other possible left-handed bases, that plays the role of the left-hand here. Our particular left hand is the basis that *co-permutes* and then *commutes* with the right hand. By *co-permute*, we mean that, $aqb = a(qb) = a(b'\hat{q})$, where, $q, a, b \in \mathbb{H}_R$, and, $b' \in \mathbb{H}_L$, so that the left-hand, b', appears when the right-hand, b, is moved around another right-hand quaternion, q, i.e., the right-hand and the left-hand *co-operate* to facilitate *permutation*; and by then *commute*, we mean that, $a(b'\hat{q}) = (ab')\hat{q} = (b'a)\hat{q} = b'a\hat{q} = ab'\hat{q}$, subsequently, in the product expressions involving several quaternions, thus subtly implying the associative law for multiplication continues to hold, despite the magical appearence of the left-hand parameter popping in somewhere in the midst of the product.

Note that there are other left hand bases, that relate differently to a given right-hand basis. For example, the conjugate basis, i^*, j^*, k^*, also forms a left-hand system, obeying the *same internal rules* as the distinguished left-hand, i', j', k'. The *intra-algebra rules* are the same. But, numbers from the left-hand conjugate basis *neither commute*, *nor co-permute*, with their corresponding right hand numbers. The *inter-algebra rules* are different. The *mixing* of the two algebras, left with right, brings out the *inter-active* distinctions between these alternate algebras that are otherwize considered to be *isomorphic algebras*.

`Second Conjugation.` The special importance of the *distinguished left-hand*, over and above all other types of alternative left-hands, in its relationship to a given right-hand, raises its status to that of a type of conjugate. As we move up the hierarchy of division algebras, $\mathbb{R} \to \mathbb{C} \to \mathbb{H} \to \mathbb{O}$, in the staircase from *reals*, *complex numbers*, *quaternions*, to *octonions*, we encounter new features in the structure of algebra. Multiplicity and complexity increases in many ways. The real numbers, $\mathbb{R}$, include the idea of *negative numbers*, and some arithmetic operations like the square root exhibit multiple roots, $(+1)^{1/2} = \pm 1$. The negative number complements the positive with an alternative. Complementary solutions to problems in real algebra exist because of the existance of these alternate roots. The complex numbers, $\mathbb{C}$, again re-iterate this theme, introducing the idea of *imaginary numbers*, and again we have multiple roots, $(-1)^{1/2} = \pm i$. This time, the idea of the negative version of the root is so important, that we raise its status, give it a special symbol * and call it *the conjugate*, $i^* = -i$. Here, the idea of *conjugation* only involves the simple operation of changing the sign. So, now the two alternate roots are, $(-1)^{1/2} = i, i^*$. These two alternate roots cover all the possibilities, and in complex numbers, solutions are either, $z = x+iy$, or, $z^* = x+i^*y,\ x, y, \in \mathbb{R}$.

When we move up to quaternions, there's quite a bit more to consider. First, we have an explosion in the number of roots; there are often no longer just two distinct alternative numbers to pick from. We may have infinitely many.

$$(-1)^{1/2} = (ai + bj + ck) \quad , a, b, c \in \mathbb{R}\ ,\ ijk = \pm 1. \tag{b-5}$$

The quaternions expand the imaginary basis units from one, i, to three, i, j, k. The conjugation * idea, from complex numbers, carries over, and again refers to a similar sign flip. However, a subtle change in the idea of the conjugate is now introduced, since in complex numbers, $(ab)^* = a^*b^*\ ,\ a, b \in \mathbb{C}$, but in quaternions we need to define, $(ab)^* = b^*a^*\ ,\ a, b \in \mathbb{H}$, instead. We must reverse the order of the product when taking the conjugate. All the other rules related to conjugation remain the same. However, the conjugate no longer suggests the unique alternate root. We have an infinity of roots, $q = (ai + bj + ck),\ a, b, c \in \mathbb{R}$, and when we take the conjugate,

$$q^* = (ai + bj + ck)^* = (ai^* + bj^* + ck^*) = (a(-i) + b(-j) + c(-k)) = ((-a)i + (-b)j + (-c)k), \tag{b-6}$$

we get back the same form of solution again. So, the conjugate, q^*, is already included among our infinite set that really represents one solution. Real numbers and complex numbers were blessed with just two complementary alternatives, but in quaternions we have infinity!

There is a sense, however, in which quaternions posess just two distinct alternatives, and that is in the choice of the triple product, $ijk = \pm 1$. Quaternions introduce the property of *handedness*, and whereas complex numbers only had *sign flips* to consider, we now have *hand flips* too. Moving up the staircase introduces new things, and new properties. There is a natural tendency to attempt to work with the principles of the lower dimensional algebra, with things familiar, and ignore the new things encountered as we move up the stairs. Understandably, this is because the complexity is increasing with every new concept or idea. But, sometimes those new things are essential to enable the work to be done on the higher level. We usually represent quaternions with one hand, the right hand, and define the triple, $ijk = -1$, to indicate this. This gives us one set of solutions to the square root, $\sqrt{-1}$.

In order to include all possible roots, we now include the other hand, the left hand, represented by, $i'j'k' = +1$, and introduce the $'$ mark to indicate this hand change. We refer to this *hand transformation* operation alternatively as *second conjugation*. The operational rules are very similar to what we're used to in conjugation, which we can also refer to now as *first conjugation*. The following table of rules shows the comparison of the two marks, * and $'$:

`1st conjugation` *	;	`2nd conjugation` $'$
$* : a \to a^*$	;	$' : a \to a'$
$(a^*)^* = a$	;	$(a')' = a$
$aa^* = a^*a$	;	$aa' = a'a$
$(ab)^* = b^*a^*$	;	$(ab)' = b'a'$
$(\lambda_1 a + \lambda_2 b)^* = \lambda_1 a^* + \lambda_2 b^*$	;	$(\lambda_1 a + \lambda_2 b)' = \lambda_1 a' + \lambda_2 b'$, $\lambda_1, \lambda_2 \in \mathbb{R}$
		(b-7)
$* : \mathbb{H}_R \to \mathbb{H}_R$, $\mathbb{H}_L \to \mathbb{H}_L$	;	$' : \mathbb{H}_R \to \mathbb{H}_L$, $\mathbb{H}_L \to \mathbb{H}_R$
$aa^* \in \mathbb{R}$, $4 \times 4 = 1$	;	$aa' \in \mathbb{X}$, $4 \times 4 = 16$
$\langle x \vert x \rangle$	;	$\vert x \rangle \langle x \vert$
$ab^* \neq b^*a$	;	$ab' = b'a$

As we move up the staircase then, we first encounter the '`sign flip`' as the `first conjugation`, in the lower dimension complex numbers. Then, up the next stair we encounter the '`hand flip`' as the `second conjugation`, in the higher dimensional quaternions. We still retain the 1st tool, but add the 2nd tool, that helps with the same jobs of finding the alternate solution and resolving algebraic expressions into simpler forms. Both conjugation and hand transformation reverse the order of products, $(ab)^* = b^*a^*$, and, $(ab)' = b'a'$. So working out problems with the hand transformation operator $'$ feels almost like using just another conjugate *. The critical exceptions are: the hand transformed parameter commutes, $ab' = b'a$ while the conjugate generally doesn't, $ab^* \neq b^*a$; and product with the conjugate contracts the number of degrees of freedom, $aa^* \in \mathbb{R}$, while product with the hand transform expands the number of degrees of freedom, $aa' \in \mathbb{X}$. This latter characteristic reminds us of a similar contrast in Quantum Physics, where the Dirac $\langle bra \vert$ and $\vert ket \rangle$ provide two product forms, $\langle x \vert x \rangle$ and $\vert x \rangle \langle x \vert$.

Apart from the separate rules for these two marks, * and $'$, there are the rules that apply when mixing the two together, and dealing with them simultaneously in expressions. For, $a, b \in \mathbb{H}_R$, we have;

$$(a')^* = (a^*)' \tag{b-8}$$
$$(ab')^* = b'^*a^* = a^*b'^* \tag{b-9}$$
$$(ab^*)' = b^{*\prime}a' \tag{b-10}$$
$$aa^* = a^*a = a'a'^* = a'^*a' = |a|^2 \in \mathbb{R} \tag{b-11}$$
$$a + a^* = a' + a'^* \in \mathbb{R} \tag{b-12}$$

The same results are obtained again, if we start with, $a, b, \in \mathbb{H}_L$, instead. Then there is the shorthand notation we introduce to facilitate working with more complicated expressions. Here we split the idea of the conjugate, $(\,)^*$, into a right conjugate operation, $(\,)^{*R}$, that attacks only the right handed quaternions, while leaving the left handed ones alone, and a left conjugate operation, $(\,)^{*L}$, that does the reverse. So, for, $a \in \mathbb{H}_R$, we have;

$$(a)^{*R} = a^* \tag{b-13}$$
$$(a)^{*L} = a \tag{b-14}$$
$$(a')^{*R} = a' \tag{b-15}$$
$$(a')^{*L} = a'^* \tag{b-16}$$

This lets us write, distinguish, evaluate and manipulate, more complex expressions like $(a, b, c \in \mathbb{H}_R)$,

$$(abc' + ca'b' + b'c'a')^{*R} = (b^*a^*c' + c^*a'b' + b'c'a') \tag{b-17}$$
$$(abc' + ca'b' + b'c'a')^{*L} = (abc'^* + cb'^*a'^* + a'^*c'^*b'^*) \tag{b-18}$$

while, without the hand qualifier, R or L, modifying the conjugate operator, $(\,)^*$, everything is conjugated thus;

$$(abc' + ca'b' + b'c'a')^* = ((abc' + ca'b' + b'c'a')^{*R})^{*L} = ((abc' + ca'b' + b'c'a')^{*L})^{*R} \tag{b-19}$$
$$= (b^*a^*c'^* + c^*b'^*a'^* + a'^*c'^*b'^*) \tag{b-20}$$

Here we must be careful to observe when the variable parameters being considered are strictly right hand or left hand in character, i.e., $a, b, c \in \mathbb{H}_R$, or $a, b, c, \in \mathbb{H}_L$. If we have mixed variables, $x = ab', y = bc', z = ca'$, then, we can't write, $(xy')^{*R} = x^* y'$. That would actually evaluate, $x^* y' = (ab')^*(bc')' = a^* b'^* cb' = a^* c|b|^2$. This is different from, $(xy')^{*R} = (ab'(bc')')^{*R} = (ab'cb')^{*R} = c^* a^* (b')^2$. We must be cautious then, when applying our one hand conjugations to mixed variables of indefinite hand.

$\hat{q}$: `The` *caret*, `the` *hat*, `and the pivot`. Most of the manipulations we do involve right handed and left handed parameters, but there's that one parameter that often sits silently around, for much of the working out in the intermediate reckoning. It helps us to initiate, and then conclude, the computations undertaken, to arrive at our ultimate results. This is our `pivot` variable, written like, $\hat{c}$. It carries that `caret` or `hat` ˆ , to mark its presence. There are some rules connected with this parameter and it's identifying mark, as can be seen in the following;

$$aq = aq\hat{} = a\hat{q} \qquad \text{where, } a, b, c, d, g, q \in \mathbb{H}_R \tag{b-21}$$

$$qa = a'q\hat{} = a'\hat{q} \tag{b-22}$$

$$aqb = a(qb)\hat{} = b'(aq)\hat{} = a(b'q\hat{}) = (ab')q\hat{} = ab'\hat{q} \tag{b-23}$$

$$(aqb + cqd)g = g'(aqb + cqd)\hat{} = g'((aqb)\hat{} + (cqd)\hat{}) = g'(b'(aq)\hat{} + d'(cq)\hat{}) = g'b'a\hat{q} + g'd'c\hat{q} = ag'b'\hat{q} + cg'd'\hat{q} \tag{b-24}$$

We can put the hat mark ˆ at the top, $\hat{q}$, or to the side of the variable, $q\hat{}$. Generally, the side hat moves in the other direction from the moving variable when initiating the change via permutation, so, $qa = a'q\hat{}$, but, $aq = \hat{}qa'$. In the matrix representation, of the two-hand quaternion algebra, the $q\hat{}$ would be the column vector representation, while, the $\hat{}q$ would be the row vector representation, and the moving quaternion, a, here becomes a square matrix, a'. We work only with the column vector equivalents[12] , however, so that there's no confusion when we simplify by placing the hat on top the variable, like, $\hat{q} \equiv q\hat{}$, for compactness. When dealing with composite expressions, like, $(aqb)\hat{}$, or when using plain ASCII text to work out problems, we use the side hat. The composite expressions are usually unnecessary, since there's always an alternative way to manipulate these expressions that avoides the requirement to know or understand all the various ˆ hat rules. However, it can be occasionaly useful to know them.

To evaluate an expression like, $aqb = b'(aq)\hat{} = b'a\hat{q} = ab'\hat{q}$, etc.., we can mentally think in terms of matrices. If, $(aq)\hat{}$, is a column vector representation of the quaternion, aq, then it is easily seen that this column vector is the same as the result, $a\hat{q}$, where, a, is the 4×4 square matrix representation of quaternion, a, while, $\hat{q}$, is the 4×1 column vector of the quaternion, q. Thinking this way, in matrices, we can readily interpret the procedures used to evaluate and uncover the hat rules in expressions like those shown above.

The idea here is not to do matrix algebra, but to *extract from* matrix algebra the key principles required *to enhance* the quaternion algebra, so that we may manipulate the quaternion symbols effectively, to arrive at solutions. But, since, in fact, we are in reality just doing matrix algebra, behind the scenes, so to speak, it can be useful to think in terms of matrices, which are presumably more familiar, to see and understand why certain rules apply. When in doubt, we can always turn to matrix algebra, to find the light. That's the beauty of the whole method.

Occasionally, we use Hamilton's 19th century notation for the six operators: S,V,K,N,T,U; the *scalar*, *vector*, *conjugate*, *norm*, *tensor*, and *versor*. (Hamilton introduced a seventh operator, M, the *mensor*, which is *the logarithm*

---

[12] We gloss over a complication here, where the square matrix representation for the left hand, a', is not identical in the two permutations, $aq = \hat{}qa'$ and $qa = a'q\hat{}$. Our restriction to column vector equivalents avoids us having to consider this subtlty. When we look more closely, we find that the distinguished left hand in row vector representation is the conjugate of the right hand in the column vector, and visa versa. That is, if, E, I, J, K, I', J', K', are the square matrix basis elements from the usual column vector approach, so that, $aq = aq\hat{}$ and $qb = b'q\hat{}$, can be represented by, $a = a_0 E + a_1 I + a_2 J + a_3 K$, $b' = b_0 E + b_1 I' + b_2 J' + b_3 K'$, then, using the same basis to express, $qa = \hat{}qa$ and $bq = \hat{}qb'$, we'd have to write, $a = a_0 E + a_1 I'^* + a_2 J'^* + a_3 K'^*$, $b' = b_0 E + b_1 I^* + b_2 J^* + b_3 K^*$. This can be seen by directly working out the expressions in components for row vector approach, as done for the column vector. Or, more readily by noting that the conjugate of a basis matrix is just its transpose, e.g, $J^T = -J = J^*$, etc. . . and then applying the usual laws of transposition for matrices, $(AB)^T = B^T A^T$, $(A + B)^T = A^T + B^T$. Here, since column and row are related by transpose, $[(aq)\hat{}]^T \equiv \hat{}(aq)$, we can evaluate the LHS: $[(aq)\hat{}]^T = [aq\hat{}]^T = [q\hat{}]^T [a]^T = \hat{}q[a]^T = \hat{}qa^*$. Then, evaluate the RHS: $\hat{}(aq) = \hat{}qa'$. This demonstrates that the distinguished left, a', in row vector, is the same as the conjugate of the right, a^*, in the column vector format.

of the tensor, $Mq = log_e(Tq)$, but felt guilty introducing so many innovations, so neglected it: pg.555, *Lectures*, 1853).

$$\begin{aligned}
q &= Sq + Vq = Tq \cdot Uq = q_0 + q_1 i + q_2 j + q_3 k && \text{(b-25)}\\
Sq &= q_0 && \text{(b-26)}\\
Vq &= q_1 i + q_2 j + q_3 k && \text{(b-27)}\\
Kq &= Sq - Vq = q_0 - q_1 i - q_2 j - q_3 k = q^* && \text{(b-28)}\\
Nq &= qKq = q_0^2 + q_1^2 + q_2^2 + q_3^2 = qq^* = |q|^2 \quad , \quad \text{pg.128, BkII, Ch.1, } \textit{Elements}\text{, 1866.} && \text{(b-29)}\\
Tq &= \sqrt{Nq} = \sqrt{q_0^2 + q_1^2 + q_2^2 = q_3^2} = |q| && \text{(b-30)}\\
Uq &= q/Tq = (q_0 + q_1 i + q_2 j + q_3 k)/\sqrt{q_0^2 + q_1^2 + q_2^2 = q_3^2} = q/|q| && \text{(b-31)}
\end{aligned}$$

Without specifically referencing the components of the quaternion, q, each of the six operators can be written soley as a sequence of basic arithmetic operations, employing any given basis, i, j, k, and the whole quaternion, q; showing that we can think of the six operators, alternatively, as six *single-variable* functions of q, instead of requiring the previous *multi-variable* expressions in the individual components, (q_0, q_1, q_2, q_3), e.g.:

$$\begin{aligned}
Sq &= S(q) = (q - iqi - jqj - kqk)/4 && \text{(b-32)}\\
Vq &= V(q) = (3q + iqi + jqj + kqk)/4 && \text{(b-33)}\\
Kq &= K(q) = -(q + iqi + jqj + kqk)/2 && \text{(b-34)}\\
Nq &= N(q) = -(q^2 + (qi)^2 + (qj)^2 + (qk)^2)/2 && \text{(b-35)}\\
Tq &= T(q) = \sqrt{(} - (q^2 + (qi)^2 + (qj)^2 + (qk)^2)/2) && \text{(b-36)}\\
Uq &= U(q) = q/\sqrt{(} - (q^2 + (qi)^2 + (qj)^2 + (qk)^2)/2) && \text{(b-37)}
\end{aligned}$$

The importance of this revelation, is that Hamilton's six dark knights, S, V, K, N, T, U, are not really primitive operators, like, $+, -, \times, \div$. They are functions, just like any other functions of a single variable. The basis units, i, j, k, are here being thought of as simply constant quaternions, of unit measure, in mutually perpendicular directions, that form the set of fixed numbers that parameterize the expressions used to formulate the functions. If we reject the idea of using any such basis numbers, i, j, k, even though they can be arbitrarly chosen, then at least one of Hamilton's operators must be considered primitive[13] ; say, S. Then, the rest follow, also without reference to any basis, as dependent on this one: $Vq = q - Sq$, $Kq = Sq - Vq$, $Nq = qKq$, $Tq = \sqrt{Nq}$, $Uq = q/Tq$. Perhaps, not surprisngly, the *conjugate*, $q^* = K(q)$, is a true function of q. Yet, owing to the arbitrary nature of the parameters, i, j, k, in that functional relation, we can often legitimately treat the pair, (q, q^*), as if they are linearly independent variables; we have exploited this trick above in solving our quadratic equations. A bit less obvious is that our new *hand transformation*, $q' = H(q)$, is also a function of q, when we extend our basis to two-hand, $(1, i, j, k, i', j', k')$. This can be easily seen, once we recognise that component extraction can also be written as functions, $q_n = X_n(q)$;

$$\begin{aligned}
q_0 &= X_0(q) = (q - iqi - jqj - kqk)/4 && \text{(b-38)}\\
q_1 &= X_1(q) = (q - iqi + jqj + kqk)/(4i) && \text{(b-39)}\\
q_2 &= X_2(q) = (q + iqi - jqj + kqk)/(4j) && \text{(b-40)}\\
q_3 &= X_3(q) = (q + iqi + jqj - kqk)/(4k) && \text{(b-41)}
\end{aligned}$$

---

[13] Modern practice is to take the conjugate * as the one primitive, $Kq = q^*$, and so express the others in terms of conjugation, as we have also done in our two-hand algebra: $Sq = (q + q^*)/2$, $Vq = (q - q^*)/2$, $Nq = qq^*$, $Tq = \sqrt{qq^*}$, $Uq = q/\sqrt{qq^*}$. But, Hamilton's *norm* and *tensor* also have additional alternate modern notations: $Nq = |q|^2$, $Tq = |q|$, the *tensor* more usually being called the "*absolute value*" or "*magnitude*" today, while the word *tensor* has taken on a more elaborate designation.

Hence, $\exists\, H : q \to q' \;\;,\;\; q \in \mathbb{H}_R, q' \in \mathbb{H}_L$, e.g.;

$$\begin{aligned} q' = H(q) \quad &= \quad [\,(q - iqi - jqj - kqk) && \text{(b-42)}\\ &\quad -ii' \cdot (q - iqi + jqj + kqk) && \text{(b-43)}\\ &\quad -jj' \cdot (q + iqi - jqj + kqk) && \text{(b-44)}\\ &\quad -kk' \cdot (q + iqi + jqj - kqk)\,]/4 && \text{(b-45)} \end{aligned}$$

$$\begin{aligned} &= \quad [\,(1 - ii' - jj' - kk') \cdot q && \text{(b-46)}\\ &\quad -(1 - ii' + jj' + kk') \cdot iqi && \text{(b-47)}\\ &\quad -(1 + ii' - jj' + kk') \cdot jqj && \text{(b-48)}\\ &\quad -(1 + ii' + jj' - kk') \cdot kqk\,]/4 && \text{(b-49)} \end{aligned}$$

Unfortunately, this hand transform function, H, that turns right hand, q, into left hand, q', is not identical to the reverse hand transform function, H^{-1}, that turns left hand into right hand, $H^{-1} : q' \to q$. The domain of H is restricted to $\mathbb{H}_R$, and it's range to $\mathbb{H}_L$, so that, $H \neq H^{-1}$. The function is not completely symmetric under the exchange of hands, $(i, j, k) \leftrightarrow (i', j', k')$. This means that, although the *conjugate* function obeys the form, $K(K(q)) = q$, consistent with the operator form, $(q^*)^* = q$, this *hand transform* function fails, $H(H(q)) \neq q$, to capture the same essential characteristic, in representing the operator form, $(q')' = q$. So, things are a bit more complicated when we extend Hamilton's one hand quaternions to a two-hand algebra.

The *hand transform* operator, $(\;)'$, is thus really more primitive than the *conjugate* operator, $(\;)^*$.

Like Hamilton's, Vq = V(q), we add, R(h) and L(h), for *right* pure quaternion and *left* pure quaternion, and, M(h), A(h), Z(h), for the *meta*, *alpha*, and *zeta* parts, thus (and not confuse our *meta*, M, with Hamilton's *mensor*!);

$$h \quad = h_0 + R(h) + L(h) + M(h) + A(h) + Z(h) \qquad \text{(b-50)}$$

$$\begin{aligned} R(h) \quad &= h_{1R}\boldsymbol{i_R} + h_{2R}\boldsymbol{j_R} + h_{3R}\boldsymbol{k_R} && \text{(b-51)}\\ L(h) \quad &= h_{1L}\boldsymbol{i_L} + h_{2L}\boldsymbol{j_L} + h_{3L}\boldsymbol{k_L} && \text{(b-52)}\\ M(h) \quad &= h_{1M}\boldsymbol{i_M} + h_{2M}\boldsymbol{j_M} + h_{3M}\boldsymbol{k_M} && \text{(b-53)}\\ A(h) \quad &= h_{1A}\boldsymbol{i_A} + h_{2A}\boldsymbol{j_A} + h_{3A}\boldsymbol{k_A} && \text{(b-54)}\\ Z(h) \quad &= h_{1Z}\boldsymbol{i_Z} + h_{2Z}\boldsymbol{j_Z} + h_{3Z}\boldsymbol{k_Z} && \text{(b-55)} \end{aligned}$$

One last thing about Hamilton's old notation, as used in our papers, concerns his differential operator $\triangleleft$. When refering to this symbol, which Hamilton called *the characteristic of operation*;

$$\triangleleft = \frac{\partial}{\partial x}i + \frac{\partial}{\partial y}j + \frac{\partial}{\partial x}k \qquad \text{(b-56)}$$

we use Hamilton's preferred "polar" form of the symbol, since it can be written both ways, $\triangleleft f$ and $f \triangleright$, implying differentiation in either one or the other direction as possible, and distinct and different, consistent with the non-abelian nature of the quaternion algebra, $ab \neq ba$, instead of Prof Tait's preferred symmetric inverted delta, ∇, called *Nabla*, which, being "non-polar" in form, makes it harder to indicate this subtle distinction. So we write,

$$\triangleleft f = \left(\frac{\partial}{\partial x}i + \frac{\partial}{\partial y}j + \frac{\partial}{\partial x}k\right) \to f \qquad\qquad f \triangleright = f \leftarrow \left(\frac{\partial}{\partial x}i + \frac{\partial}{\partial y}j + \frac{\partial}{\partial x}k\right) \qquad \text{(b-57)}$$

interpreting Hamilton's intention, to use the polar symbol as such. The effect is to essentially obtain, + *curl*, in the one, and, − *curl*, in the other, when the expressions are evaluated; all other parts of the differential expression being otherwise the same.

`Complexifying`. While, $\mathbb{X}$, is able to represent any 4×4 matrix with real entries, $\mathbb{R}$, this is not sufficient for modern quantum physics. So, to facilitate the extension of the algebra to handle the 4×4 matrices over the complex number field, $\mathbb{C}$, as used in physics, we define the following symbol, $\ddot{\mathbb{X}}$,

$$\ddot{\mathbb{X}} = \{h \mid h = \sum_j c_j e_j \;;\; e_j \in \mathbb{X}_b^+, c_j \in \mathbb{C}\} \qquad \text{(b-58)}$$

The double dot ¨ indicates we've doubled the number of degrees of freedom in our, $\mathbb{X}$. In this case, to distinguish the complex number imaginary unit from Hamilton's quaternion units, we use the 'iota' symbol, $\iota, with, \iota^2 = -1$, and write, $z = x + \iota y \in \mathbb{C}$, reserving, i, for the quaternions. It is then undertstood that ι commutes with i, j, k, and, i', j', k', and conjugating the complexified quaternions does not automatically conjugate the complex number coefficients. The two conjugation operations must be treated separately. We won't say more about this here, except to note that, when not dealing with, $\ddot{\mathbb{X}}$, we often reuse the 'iota' symbol, ι, in other contexts, where it does not necessarily commute with all, i, j, k, but usually because it's something like, $\iota = n_1 i + n_2 j + n_3 k,\ n_1, n_2, n_3 \in \mathbb{R}, (n_1^2 + n_2^2 + n_3^2) = 1$, so that, $(\iota)^2 = -1$, still effectively represents a complex number imaginary in some way, like subplane $\mathbb{C} \subset \mathbb{H}$.

Now, we have used this double dot ¨ before, in other contexts, so we'll just mention the alternative uses here, to alert the reader to our multiple usage. In a previous paper, we use the double dot to represent a "split" algebra, e.g. $\ddot{\mathbb{H}}$ is the split variation of Hamilton's quaternions $\mathbb{H}$; then, $\ddot{\mathbb{O}}$ is the split version of regular Octonions $\mathbb{O}$; while, $\ddot{\tilde{\mathbb{O}}}$ is both *split* and *twisted* algebra related to the Octonions. So, we have the various algebras: $\mathbb{R}, \mathbb{C}, \ddot{\mathbb{C}}, \mathbb{H}, \ddot{\mathbb{H}}, \mathbb{O}, \ddot{\mathbb{O}}, \ddot{\tilde{\mathbb{O}}}$, and a few product algebras, $\ddot{\mathbb{O}} \times \mathbb{O} \cong M_{[\times]}(8, \mathbb{R}),\ \ddot{\mathbb{H}} \times \mathbb{O} \cong M_{[\times]}(4, \mathbb{C}),\ \mathbb{O} \times \mathbb{C} \cong M_{[\times]}(2, \mathbb{H})$, met so far in our papers.

We also used, $\ddot{h}_n = a_1 b'_1 + a_2 b'_2 + \cdots + a_n b'_n,\ \ a_k \in \mathbb{H}_R, b'_k \in \mathbb{H}_L$, where, in this context, the double dot ¨ indicates the hexpe number, h, is being specifically given in bilateral format, with right-hand times left-hand quaternion terms, $a_k b'_k$, with full quaternions, as opposed to, for example, a basis element format, $h = \sum c_k e_k,\ \ c_k \in \mathbb{R}, e_k \in \mathbb{X}_b^+$.

`Reference Video Notation`. In the reference video, of APPENDIX A, we use an ordinary text editor to present the calculations. Since we're limited to the ASCII character set, we have to adopt some convenient conventions. In general, when working in, or exhibiting work, using ASCII text, depending on context, we'll substitute specific english letters for the greek letters, and modify the placement of symbolic marks that makeup expressions.

The real valued parameter λ is represented by the letter `r`, and for, γ, we use `g`.

The following symbols are used to write expressions in ASCII text.

```
+   -   .     \   /   *   '   ^   ( )  [ ]   { }   | |   ,    =
```

Add and subtract are the usual, `a + b`, and, `a - b`, while, multiply is usually simply justaposition, `ab`, only occasionally using the dot, `a.b`, to represent multiplication in circumstances where it makes things more clear. We often like to separate the right hand variables from the left hand using the dot, like, `ab.c'd'`, to see the distinctions more easily. We also like to introduce, and then remove, the dot, during manipulations, to help clarify what is being changed, like, `abc*'c'b'` $\rightarrow$ `ab.c*'c'.b'` $\rightarrow$ `ab.|c*'c'|.b'` $\rightarrow$ `ab.|c|^2.b` $\rightarrow$ `|c|^2.abb'`. The dot is thus used as an optional, but convenient mark. Division is left or right, `a\b` or `a/b`, but alternatively we write, `a^-1.b` and `ba^-1`, or sometimes, `a^(-1)b` and `ba^(-1)`. Ocassionally, when there's a long expression in a fraction, we'll draw a long line like `----------` and put the numerator expression above, and denomonator expression below on separate lines. The `*` mark is our "conjugation," which we also call 1st conjugation. So, `c*b`, means, `(c*)(b)`, i.e., $c^* \cdot b$, and *not* `c` multiplied by `b`, as otherwise used in MATLAB and other computer code. Similarly, our 2nd conjugation mark `'` refers to "hand transformation," and not to be confused with MATLAB's use for hermitian conjugate and transposition. The circumflex mark `^` is used in two different contexts. The first is to raise a variable to some power, like, `a^2`, for `a`-squared. The second is for our pivot variable "hat" mark, which we can't place on top the variable here, so it goes to the right side of the parameter, like, `q^` . Since nothing else is allowed on the right side of a pivot variable, there's no confusion in the dual use of this mark. As soon as a left hand variable is moved over to the right of the pivot, the hat is removed, `a'b^` $\rightarrow$ `a'.b^` $\rightarrow$ `b.a` $\rightarrow$ `ba`. The parenthesis and square brackets, `( )` and `[ ]`, are used similarly to group sub-expressions. But, the `[ ]` are generally introduced "temporarily" to indicate that something within is about to change, when manipulating expressions. While, `( )` are the choice for representing the final result of those changes. Thus, the square `[ ]` and the dot `.` are used similarly, to indicate the active state of expressions during manipulation, so the current focus of the calculation is always clear. An alternate use of the square brackets `[,]` occurs when we specifically include the comma " , " within, which then represents the commutator, `[a,b] = ab - ba`, and similarly the curly braces {,} with its comma " , " represents the corresponding anti-commutator, `{a,b} = ab + ba`. The bars | | are the usual absolute value symbols, indicating the magnitude, `|a| = sqrt(a0^2 + a1^2 + a2^2 + a3^2)`, and, of course, the equal sign `=` has the usual meaning.

Wherever possible, the variable letter names are used exactly as in this paper itself, so that the correspondences can be readily seen.

Having said all this, given the large number of computations involved, some of these conventions are treated as optional, especially, when skipping over a few intermediate steps, to move ahead in the calculation more quickly. We'll usually start out by being very detailed, showing every step of the moves, but once the viewer has seen something like it before, we'll often elect to jump over some inbetween expression manipulations.

APPENDIX C: One-Step Quadratics

Most of the discussion in this paper deals with quadratic equations that specifically have non-vanishing linear terms (the $b \neq 0$, requirement). Here, for completeness, we illustrate those other equations that are typically solvable with just a *one-step* method, i.e. when, $b = 0$. The following equations all have the same easily determined *magnitude*, $\lambda = \sqrt{|c|/|a|}$, anyway, so essentially, all the derivation can be seen to be concerned with finding the suitable *directions*, to complement this known *magnitude*. To keep everything consistent, with things looking familiar, we set out, as before, with separated parameters; $q = \lambda p,\ \ p, q \in \mathbb{H}_R,\ |p| = 1,\ \lambda \in \mathbb{R},\ \lambda \geq 0$. For the first equation `(b=0)`,

`#1:` `aqq + c = 0`

$$aq^2 + c = 0 \tag{c-1}$$
$$aq^2 = -c \tag{c-2}$$
$$q^2 = -a^{-1}c \tag{c-3}$$
$$= -\frac{a^*c}{|a|^2} \tag{c-4}$$

$$\text{Let, } q = \lambda p \ , \qquad p, q \in \mathbb{H}_R,\ |p| = 1,\ \lambda \in \mathbb{R},\ \lambda \geq 0 \tag{c-5}$$

$$\lambda^2 = |q|^2 = |q^2| = \left|-\frac{a^*c}{|a|^2}\right| = \frac{|a^*c|}{|a|^2} = \frac{|a^*||c|}{|a|^2} = \frac{|c|}{|a|} \tag{c-6}$$

$$\lambda = \sqrt{\frac{|c|}{|a|}} \tag{c-7}$$

$$p^2 = \frac{q^2}{\lambda^2} = -\frac{a^*c}{|a|^2}\frac{|a|}{|c|} = -\frac{a^*c}{|a||c|} = -\frac{a^*c}{|ac|} = -\frac{a^*c}{|a^*c|} \tag{c-8}$$

.

`Case 1:` $a^*c \in \mathbb{R},\ a^*c > 0.$
`Case 2:` $a^*c \in \mathbb{R},\ a^*c < 0.$
`Case 3:` $a^*c \notin \mathbb{R},\ S(a^*c) = 0.$
`Case 4:` $a^*c \notin \mathbb{R},\ S(a^*c) \neq 0.$

Note that $a^*c = 0$, implies, $a = 0$ or $c = 0$. But, if, $a = 0$, there's no quadratic equation, so we exclude it; and if, $c = 0$, then, trivially, $q = 0$, so we exclude it also.

In the `1st case`, since, $a^*c \in \mathbb{R}$, and, $a^*c > 0$, then, $a^*c = |a^*c|$, so, $a^*c/|a^*c| = +1$, and it follows that the square of the direction unit is negative, $p^2 = -1$, which has an infinite set of solutions, given by,

$$p = (n_1 i + n_2 j + n_3 k) \ , \quad n_1, n_2, n_3 \in \mathbb{R},\ n_1^2 + n_2^2 + n_3^2 = 1. \tag{c-9}$$
$$\lambda = \sqrt{|c|/|a|} \tag{c-10}$$
$$q = \lambda p = \sqrt{|c|/|a|} \cdot (n_1 i + n_2 j + n_3 k) \tag{c-11}$$

In the `2nd case`, since, $a^*c \in \mathbb{R}$, and, $a^*c < 0$, then, $a^*c = -|a^*c|$, so, $a^*c/|a^*c| = -1$, and it follows that the square of the direction unit is positive, $p^2 = +1$, which has two solutions [14] , given by,

$$p = \pm 1 \tag{c-12}$$
$$\lambda = \sqrt{|c|/|a|} \tag{c-13}$$
$$q = \lambda p = \sqrt{|c|/|a|} \cdot (\pm 1) \tag{c-14}$$

---

[14] Although, $p^2 = +1$, could be seen to have, not only the two real value solutions, $p = \pm 1$, which we indicate here, but also imaginary solutions among the commutative hypercomplex M, A, Z numbers! Then, $p = \pm 1, \pm i_M, \pm j_M, \pm k_M, \pm i_A, \pm j_A, \pm k_A, \pm i_Z, \pm j_Z, \pm k_Z$, which is just another way of writing, $p = \pm 1, \pm ii', \pm jj', \pm kk', \pm jk', \pm ki', \pm ij', \pm kj', \pm ik', \pm jk'$; a total of 20 solutions, for $p \in \mathbb{X}$. However, we are only interested in right-hand quaternion solutions, $p \in \mathbb{H}_R \subset \mathbb{X}$, where there are only the two real valued ± 1 results. Similarily, $p^2 = -1,\ \Longrightarrow\ \ p = (n_1 i + n_2 j + n_3 k)$ or $p = (m_1 i' + m_2 j' + m_3 k')$, but we want, $p \in \mathbb{H}_R$, so exclude the left hand.

In the `3rd case`, since, $a^*c \notin \mathbb{R}$, but, $S(a^*c) = 0$, the square of the unit direction is a *pure quaternion*, with no scalar part, $p^2 = Vp^2$, so, if we write out the components, and compare terms,

$$\begin{aligned} p &= p_0 + p_1 i + p_2 j + p_3 k && \text{(c-15)} \\ p^2 &= p_0^2 - p_1^2 - p_2^2 - p_3^2 + 2p_0(p_1 i + p_2 j + p_3 k) && \text{(c-16)} \end{aligned}$$

$$\begin{aligned} p_0^2 - p_1^2 - p_2^2 - p_3^2 &= 0 && \text{(c-17)} \\ 2p_0(p_1 i + p_2 j + p_3 k) &= -\frac{a^*c}{|a^*c|} = p^2 && \text{(c-18)} \end{aligned}$$

$$1 = |p|^2 = |p^2| = |2p_0(p_1 i + p_2 j + p_3 k)|^2 = 4p_0^2(p_1^2 + p_2^2 + p_3^2) = 4p_0^2(p_0^2) = (2p_0^2)^2 \qquad \text{(c-19)}$$

$$\begin{aligned} 2p_0^2 &= \pm 1 \ , \quad \text{but, } p_0 \in \mathbb{R} \quad \therefore \ 2p_0^2 = +1 && \text{(c-20)} \\ p_0 &= \pm\frac{1}{\sqrt{2}} && \text{(c-21)} \\ (p_1 i + p_2 j + p_3 k) &= -\frac{1}{2p_0} \cdot \frac{a^*c}{|a^*c|} = \left(\pm\frac{1}{\sqrt{2}}\right) \cdot \left(-\frac{a^*c}{|a^*c|}\right) && \text{(c-22)} \end{aligned}$$

$$\begin{aligned} p &= \left(\pm\frac{1}{\sqrt{2}}\right) \cdot \left(1 - \frac{a^*c}{|a^*c|}\right) && \text{(c-23)} \\ \lambda &= \sqrt{|c|/|a|} && \text{(c-24)} \\ q &= \lambda p = \pm\sqrt{\frac{|c|}{2|a|}} \cdot \left(1 - \frac{a^*c}{|a^*c|}\right) && \text{(c-25)} \end{aligned}$$

In the `4th case`, since, $a^*c \notin \mathbb{R}$, and, $S(a^*c) \neq 0$, the square of the unit direction is now a *full quaternion*, with non-zero scalar part, $p^2 = Sp^2 + Vp^2$, so, lets introduce a new quaternion variable, $p^2 = M = M_0 + M_1 i + M_2 j + M_3 k$.

$$\begin{aligned} p^2 &= -\frac{a^*c}{|a^*c|} = M_0 + M_1 i + M_2 j + M_3 k && \text{(c-26)} \\ |p|^2 &= 1, \ \therefore \quad |p^2|^2 = 1, \quad \therefore \quad M_0^2 + M_1^2 + M_2^2 + M_3^2 = 1 && \text{(c-27)} \end{aligned}$$

$$\begin{aligned} p_0^2 - p_1^2 - p_2^2 - p_3^2 &= M_0 && \text{(c-28)} \\ 2p_0 \cdot (p_1 i + p_2 j + p_3 k) &= (M_1 i + M_2 j + M_3 k) && \text{(c-29)} \end{aligned}$$

If $p_0 = 0$, then, $M_1 = M_2 = M_3 = 0$, and, $M = M_0$ is Real; but, $a^*c \notin \mathbb{R}$, here, so we exclude this.

$$\therefore \quad p_0 \neq 0 \qquad \text{(c-30)}$$

$$\begin{aligned} (p_1 i + p_2 j + p_3 k) &= (M_1 i + M_2 j + M_3 k)/(2p_0) && \text{(c-31)} \\ p_0^2 - (M_1^2 + M_2^2 + M_3^2)/(4p_0^2) &= M_0 && \text{(c-32)} \\ M_1^2 + M_2^2 + M_3^2 &= 1 - M_0^2 && \text{(c-33)} \\ p_0^2 - (1 - M_0^2)/(4p_0^2) &= M_0 && \text{(c-34)} \\ p_0^4 - (1 - M_0^2)/4 &= M_0 p_0^2 && \text{(c-35)} \\ p_0^4 - M_0 p_0^2 - (1 - M_0^2)/4 &= 0 && \text{(c-36)} \end{aligned}$$

$$(p_0^2 - M_0/2)^2 - M_0^2/4 - (1 - M_0^2)/4 = 0 \tag{c-37}$$

$$(p_0^2 - M_0/2)^2 = 1/4 \tag{c-38}$$

$$p_0^2 - M_0/2 = \pm 1/2 \tag{c-39}$$

$$p_0^2 = M_0/2 \pm 1/2 = (M_0 \pm 1)/2 \tag{c-40}$$

$$p_0 = \pm\sqrt{(M_0 \pm 1)/2}\ , \qquad p_0, M_0 \in \mathbb{R} \tag{c-41}$$

Because, $|M| \geq |M_0|$, we have, $|M_0| \leq 1$, so, $M_0 + 1$, is the only option, since the $M_0 - 1 < 0$, for $M_0 > 0$, or $M_0 < 0$.

$$p_0 = \pm\sqrt{(M_0 + 1)/2} \tag{c-42}$$

$$p = \pm\left[\sqrt{(M_0 + 1)/2} + (M_1 i + M_2 j + M_3 k)/(2\sqrt{(M_0 + 1)/2})\right] \tag{c-43}$$

$$p = \pm\frac{1}{2}\frac{M_0 + 1 + (M_1 i + M_2 j + M_3 k)}{\sqrt{(M_0 + 1)/2}} \quad = \quad \pm\frac{1 + M}{2\sqrt{(M_0 + 1)/2}} \tag{c-44}$$

putting in,

$$M = -\frac{a^* c}{|a^* c|}, \qquad \texttt{and,} \qquad M_0 \quad = S\left(-\frac{a^* c}{|a^* c|}\right) = -\frac{(a^* c) + (a^* c)^*}{2|a^* c|} \tag{c-45}$$

we have,

$$p \quad = \quad \pm\frac{1 - (a^* c)/|a^* c|}{\sqrt{2 - (a^* c + (a^* c)^*)/|a^* c|}} \tag{c-46}$$

$$\lambda \quad = \quad \sqrt{|c|/|a|} \tag{c-47}$$

$$q \quad = \quad \lambda p \quad = \quad \pm\sqrt{\frac{|c|}{|a|}} \cdot \frac{1 - (a^* c)/|a^* c|}{\sqrt{2 - (a^* c + (a^* c)^*)/|a^* c|}} \tag{c-48}$$

For the second quadratic equation (b=0), we again set; $q = \lambda p,\ \ p, q \in \mathbb{H}_R,\ |p| = 1,\ \lambda \in \mathbb{R},\ \lambda \geq 0.$

#2: qqa + c = 0

$$q^2 a + c = 0 \tag{c-49}$$
$$q^2 a = -c \tag{c-50}$$
$$q^2 = -ca^{-1} \tag{c-51}$$
$$= -\frac{ca^*}{|a|^2} \tag{c-52}$$

$$\text{Let, } q = \lambda p \ , \qquad p, q \in \mathbb{H}_R,\ |p| = 1,\ \lambda \in \mathbb{R},\ \lambda \geq 0 \tag{c-53}$$

$$\lambda^2 = |q|^2 = |q^2| = \left| -\frac{ca^*}{|a|^2} \right| = \frac{|ca^*|}{|a|^2} = \frac{|c||a^*|}{|a|^2} = \frac{|c|}{|a|} \tag{c-54}$$

$$\lambda = \sqrt{\frac{|c|}{|a|}} \tag{c-55}$$

$$p^2 = \frac{q^2}{\lambda^2} = -\frac{ca^*}{|a|^2}\frac{|a|}{|c|} = -\frac{ca^*}{|a||c|} = -\frac{ca^*}{|ca|} = -\frac{ca^*}{|ca^*|} \tag{c-56}$$

.

Case 1: $ca^* \in \mathbb{R},\ ca^* > 0.$
Case 2: $ca^* \in \mathbb{R},\ ca^* < 0.$
Case 3: $ca^* \notin \mathbb{R},\ S(ca^*) = 0.$
Case 4: $ca^* \notin \mathbb{R},\ S(ca^*) \neq 0.$

Note that $ca^* = 0$, implies, $a = 0$ or $c = 0$. But, if, $a = 0$, there's no quadratic equation, so we exclude it; and if, $c = 0$, then, trivially, $q = 0$, so we exclude it also. Notice also, that things are very similar here in the #2 eqn to that above in the #1 eqn. We only have to substitute ca^* for the previous a^*c expression, to find our new solutions, since the steps of derivation are essentially identical. So, we can immediately write down the solutions here.

In the 1st case, since, $ca^* \in \mathbb{R}$, and, $ca^* > 0$, then, $ca^* = |ca^*|$, so, $ca^*/|ca^*| = +1$, and it follows that the square of the direction unit is negative, $p^2 = -1$, which has an infinite set of solutions, given by,

$$p = (n_1 i + n_2 j + n_3 k) \ , \quad n_1, n_2, n_3 \in \mathbb{R},\ n_1^2 + n_2^2 + n_3^2 = 1. \tag{c-57}$$
$$\lambda = \sqrt{|c|/|a|} \tag{c-58}$$
$$q = \lambda p = \sqrt{|c|/|a|} \cdot (n_1 i + n_2 j + n_3 k) \tag{c-59}$$

In the 2nd case, since, $ca^* \in \mathbb{R}$, and, $ca^* < 0$, then, $ca^* = -|ca^*|$, so, $ca^*/|ca^*| = -1$, and it follows that the square of the direction unit is positive, $p^2 = +1$, which has two solutions, given by,

$$p = \pm 1 \tag{c-60}$$
$$\lambda = \sqrt{|c|/|a|} \tag{c-61}$$
$$q = \lambda p = \sqrt{|c|/|a|} \cdot (\pm 1) \tag{c-62}$$

In the 3rd case, since, $ca^* \notin \mathbb{R}$, but, $S(ca^*) = 0$, the square of the unit direction is a *pure quaternion*, with no scalar part, $p^2 = Vp^2$, and, making the substitution ca^* for the a^*c in the previous corresponding case of eqn #1, we immediately have the solution;

$$p = \left(\pm \frac{1}{\sqrt{2}} \right) \cdot \left(1 - \frac{ca^*}{|ca^*|} \right) \tag{c-63}$$
$$\lambda = \sqrt{|c|/|a|} \tag{c-64}$$
$$q = \lambda p = \pm \sqrt{\frac{|c|}{2|a|}} \cdot \left(1 - \frac{ca^*}{|ca^*|} \right) \tag{c-65}$$

In the `4th case`, since, $ca^* \notin \mathbb{R}$, and, $S(ca^*) \neq 0$, the square of the unit direction is now a *full quaternion*, with non-zero scalar part, $p^2 = Sp^2 + Vp^2$, and, making the substitution ca^* for the a^*c in the previous corresponding case of eqn #1, we immediately have the solution;

$$p = \pm \frac{1 - (ca^*)/|ca^*|}{\sqrt{2 - (ca^* + (ca^*)^*)/|ca^*|}} \tag{c-66}$$

$$\lambda = \sqrt{|c|/|a|} \tag{c-67}$$

$$q = \lambda p = \pm \sqrt{\frac{|c|}{|a|}} \cdot \frac{1 - (ca^*)/|ca^*|}{\sqrt{2 - (ca^* + (ca^*)^*)/|ca^*|}} \tag{c-68}$$

For the third quatratic equation (`b=0`), we set, $q = \lambda p,\ \ p, q \in \mathbb{H}_R,\ |p| = 1,\ \lambda \in \mathbb{R},\ \lambda \geq 0.$

`#3:` `qaq + c = 0`

$$qaq + c = 0 \tag{c-69}$$
$$qaq = -c \tag{c-70}$$
$$aqaq = -ac \tag{c-71}$$
$$(aq)^2 = -ac \tag{c-72}$$

$$\text{Let, } q = \lambda p\ \ , \quad P = ap/|a|\ \ , \qquad p, q, P \in \mathbb{H}_R,\ |p| = 1,\ |P| = 1,\ \lambda \in \mathbb{R},\ \lambda \geq 0 \tag{c-73}$$

$$\lambda^2 = |q|^2 = \frac{|q||a||q|}{|a|} = \frac{|qaq|}{|a|} = \frac{|-c|}{|a|} = \frac{|c|}{|a|} \tag{c-74}$$

$$\lambda = \sqrt{\frac{|c|}{|a|}} \tag{c-75}$$

$$P^2 = \frac{(ap)^2}{|a|^2} = \frac{(aq)^2}{|a|^2\lambda^2} = -\frac{ac}{|a|^2}\frac{|a|}{|c|} = -\frac{ac}{|a||c|} = -\frac{ac}{|ac|} \tag{c-76}$$

.

`Case 1:` $ac \in \mathbb{R},\ ac > 0.$
`Case 2:` $ac \in \mathbb{R},\ ac < 0.$
`Case 3:` $ac \notin \mathbb{R},\ S(ac) = 0.$
`Case 4:` $ac \notin \mathbb{R},\ S(ac) \neq 0.$

Note that $ac = 0$, implies, $a = 0$ or $c = 0$. But, if, $a = 0$, there's no quadratic equation, so we exclude it; and if, $c = 0$, then, trivially, $q = 0$, so we exclude it also. Things are somewhat similar to the #1 and #2 eqns. But, here in the #3 eqn, we have a little twist. We effectively have the equation, $Q^2 + C = 0$, with, $Q = aq$, and, $C = ac$. We can think of this as the form, $AQ^2 + C = 0$, with, $A = 1$, and thus use the #1 eqn solution, or think of it as the form, $Q^2A + C = 0$, with, $A = 1$, and use the #2 eqn solution. Having found the, Q, we then immediately have, $q = a^{-1}Q$. Similar to, $q = \lambda p$, we can set, $Q = \Lambda P$. Then, $\Lambda = |Q| = |aq| = |a|\lambda$, and, $P = Q/\Lambda = aq/(|a|\lambda) = ap/|a|$. Thus, we can re-use the previous solution results, without re-working all the steps of derivation again, using appropriate substitutions. So, let's see what the solutions look like.

In the `1st case`, since, $ac \in \mathbb{R}$, and, $ac > 0$, then, $ac = |ac|$, so, $ac/|ac| = +1$, and it follows that the square of the direction unit is negative, $P^2 = -1$, which has an infinite set of solutions, from which, $p = |a|a^{-1}P = a^*P/|a|$, follows, and the solutions are therefore given by,

$$p = \frac{a^*}{|a|} \cdot (n_1 i + n_2 j + n_3 k)\ \ , \quad n_1, n_2, n_3 \in \mathbb{R},\ n_1^2 + n_2^2 + n_3^2 = 1. \tag{c-77}$$
$$\lambda = \sqrt{|c|/|a|} \tag{c-78}$$
$$q = \lambda p = \sqrt{|c|/|a|} \cdot \frac{a^*}{|a|} \cdot (n_1 i + n_2 j + n_3 k) \tag{c-79}$$

In the `2nd case`, since, $ac \in \mathbb{R}$, and, $ac < 0$, then, $ac = -|ac|$, so, $ac/|ac| = -1$, and it follows that the square of the direction unit is positive, $P^2 = +1$, which has two solutions, from which, $p = a^*P/|a|$, follows, and the solutions are therefore,

$$p = \frac{a^*}{|a|} \cdot (\pm 1) \tag{c-80}$$
$$\lambda = \sqrt{|c|/|a|} \tag{c-81}$$
$$q = \lambda p = \sqrt{|c|/|a|} \cdot \frac{a^*}{|a|} \cdot (\pm 1) \tag{c-82}$$

In the `3rd case`, since, $ac \notin \mathbb{R}$, but, $S(ac) = 0$, the square of the unit direction is a *pure quaternion*, with no scalar part, $P^2 = VP^2$, so, we can work through the case in a similar manner as before in #1, thus replacing a^*c there with the current ac equivalent, then making the adjustment for, $p = a^*P/|a|$, this lets us immediately write,

$$p = \left(\pm\frac{1}{\sqrt{2}}\right) \cdot \frac{a^*}{|a|} \cdot \left(1 - \frac{ac}{|ac|}\right) \tag{c-83}$$

$$\lambda = \sqrt{|c|/|a|} \tag{c-84}$$

$$q = \lambda p = \pm\sqrt{\frac{|c|}{2|a|}} \cdot \frac{a^*}{|a|} \cdot \left(1 - \frac{ac}{|ac|}\right) \tag{c-85}$$

In the `4th case`, since, $ac \notin \mathbb{R}$, and, $S(ac) \neq 0$, the square of the unit direction is now a *full quaternion*, with non-zero scalar part, $P^2 = SP^2 + VP^2$, this is again similar to the #1 eqn corresponding case, and, once again, we only need to replace the previous a^*c with our current ac expression, then make the adjustment for, $p = a^*P/|a|$, and the solution here is thus,

$$p = \pm\frac{a^*}{|a|} \cdot \frac{1 - (ac)/|ac|}{\sqrt{2 - (ac + (ac)^*)/|ac|}} \tag{c-86}$$

$$\lambda = \sqrt{|c|/|a|} \tag{c-87}$$

$$q = \lambda p = \pm\sqrt{\frac{|c|}{|a|}} \cdot \frac{a^*}{|a|} \cdot \frac{1 - (ac)/|ac|}{\sqrt{2 - (ac + (ac)^*)/|ac|}} \tag{c-88}$$

We could also write this #3 eqn, $qaqa + ca = 0$, and work with the expression ca instead of the ac which we just used. Then, the factor, $a^*/|a|$, must appear on the right of the expressions, instead of the left, as seen above. The four cases would then have corresponding solutions, in ca form, given by,

`Case 1:` $ca \in \mathbb{R},\ ca > 0$: $\quad q = \sqrt{|c|/|a|} \cdot (n_1 i + n_2 j + n_3 k) \cdot a^*/|a|$ (c-89)

`Case 2:` $ca \in \mathbb{R},\ ca < 0$: $\quad q = \pm\sqrt{|c|/|a|} \cdot a^*/|a|$ (c-90)

`Case 3:` $ca \notin \mathbb{R},\ S(ca) = 0$: $\quad q = \pm\sqrt{|c|/(2|a|)} \cdot (1 - ca/|ca|) \cdot a^*/|a|$ (c-91)

`Case 4:` $ca \notin \mathbb{R},\ S(ca) \neq 0$: $\quad q = \pm\sqrt{|c|/|a|} \cdot [\,(1 - ca/|ca|)/\sqrt{2 - (ca + (ca)^*)/|ca|}\,] \cdot a^*/|a|$ (c-92)

For the fourth quadratic equation `(b=0)`:

`#4:` `qaq + c = 0` just see #3 above. (c-93)

For the fifth quadratic equation `(b=0)`:

`#5:` `aqq + c = 0` just see #1 above. (c-94)

For the sixth quadratic equation `(b=0)`:

`#6:` `qqa + c = 0` just see #2 above. (c-95)

For the seventh quadratic equations (b=0):

$$\texttt{\#7-1:} \qquad \sum_{j=1,n} a_{j,1}\cdot q\cdot a_{j,2}\cdot q + c = 0 \tag{c-96}$$

If this equation has any solution, then there must be a set of constant quaternion numbers, $c_j,\ j=1,..,n$, such that, $\sum_{j=1,n} c_j = c$, and, $a_{j,1}\cdot q\cdot a_{j,2}\cdot q + c_j = 0$, for each index, j. So, we can consider each term independently of the other and solve the n equations;

$$a_{j,1}\cdot q\cdot a_{j,2}\cdot q + c_j = 0 \tag{c-97}$$

But this would potentially give us n different values for, q. So, we then adjust the arbitrary constants, c_j, until all these equations give the same value for, q. First, transform these equations,

$$q\cdot a_{j,2}\cdot q + a_{j,1}^{-1}\cdot c_j = 0 \tag{c-98}$$

These $j=1..n$ equations all have the form, $qaq+c=0$, which we know how to solve. We just solved this above, giving the 'ca' form of solutions:

$$\texttt{Case 1:}\quad ca\in\mathbb{R},\ ca>0: \qquad q = \sqrt{|c|/|a|}\cdot(n_1 i + n_2 j + n_3 k)\cdot a^*/|a| \tag{c-99}$$

$$\texttt{Case 2:}\quad ca\in\mathbb{R},\ ca<0: \qquad q = \pm\sqrt{|c|/|a|}\cdot a^*/|a| \tag{c-100}$$

$$\texttt{Case 3:}\quad ca\notin\mathbb{R},\ S(ca)=0: \qquad q = \pm\sqrt{|c|/(2|a|)}\cdot(1-ca/|ca|)\cdot a^*/|a| \tag{c-101}$$

$$\texttt{Case 4:}\quad ca\notin\mathbb{R},\ S(ca)\neq 0: \qquad q = \pm\sqrt{|c|/|a|}\cdot[\,(1-ca/|ca|)/\sqrt{2-(ca+(ca)^*)/|ca|}\,]\cdot a^*/|a| \tag{c-102}$$

So, now we need to replace, $c\to a_{j,1}^{-1}c_j,\ a\to a_{j,2}$, to get each solution, q_j, for the n equations. Then, find the c_j, such that all q are the same, which is done by simply equating the results, $q_j = q_k,\ \forall j,k = 1...n$. Using the fact that, $|q| = \sqrt{|c|/|a|}$, when we replace the coefficients, and apply the equality, $|q_j| = |q_k|$, we arive at the relation,

$$\frac{|c_j|}{|a_{j,1}||a_{j,2}|} = \frac{|c_k|}{|a_{k,1}||a_{k,2}|} = \gamma^2 \qquad ,\gamma\in\mathbb{R} \tag{c-103}$$

Thus all the coefficients have magnitudes given by some fixed scalar, γ^2, and the relevant $a_{j,1}, a_{j,2}$ coefficients;

$$|c_j| \quad = \gamma^2\cdot|a_{j,1}||a_{j,2}| \tag{c-104}$$

$$c_j \quad = \gamma^2\cdot|a_{j,1}||a_{j,2}|\cdot u_j\ , \qquad \text{where,}\ \ |u_j| = 1 \tag{c-105}$$

Then the quaternions, q_j, all have the form

$$q_j = \gamma\cdot u \tag{c-106}$$

Thus,

$$\gamma\cdot u\cdot a_{j,2}\cdot\gamma\cdot u + a_{j,1}^{-1}\cdot c_j = 0 \tag{c-107}$$

$$c_j = -a_{j,1}\cdot\gamma\cdot u\cdot a_{j,2}\cdot\gamma\cdot u \tag{c-108}$$

$$c = \sum_{j=1,n} c_j = -\gamma^2\cdot\sum_{j=1,n} a_{j,1}\cdot u\cdot a_{j,2}\cdot u \tag{c-109}$$

$$\gamma^2 = \frac{|c|}{|\sum_{j=1,n} a_{j,1}\cdot u\cdot a_{j,2}\cdot u|} \tag{c-110}$$

$$\texttt{Let,}\quad w_j = a_{j,1}^{-1}c_j a_{j,2} \tag{c-111}$$

$$\texttt{Case 1:}\quad w_j\in\mathbb{R},\ w_j>0: \qquad u = (n_1 i + n_2 j + n_3 k)\cdot a_{j,2}^*/|a_{j,2}| \tag{c-112}$$

$$\texttt{Case 2:}\quad w_j\in\mathbb{R},\ w_j<0: \qquad u = \pm a_{j,2}^*/|a_{j,2}| \tag{c-113}$$

$$\texttt{Case 3:}\quad w_j\notin\mathbb{R},\ S(w_j)=0: \qquad u = \pm[\,(1-u_j a_{j,2}/|a_{j,2}|)/\sqrt{2}\,]\cdot a_{j,2}^*/|a_{j,2}| \tag{c-114}$$

$$\texttt{Case 4:}\quad w_j\notin\mathbb{R},\ S(w_j)\neq 0: \qquad u = \pm[\,(1-u_j a_{j,2}/|a_{j,2}|)/\sqrt{2-(u_j a_{j,2}+(u_j a_{j,2})^*)/|a_{j,2}|}\,]\cdot a_{j,2}^*/|a_{j,2}| \tag{c-115}$$

APPENDIX D: One-hand derivation method – using the Right-Hand Only

$\underline{aq^2+bq+c=0}$: Consider...Quaternion Quadratic Equation #1:

$$aqq+bq+c=0 \tag{d-1}$$
$$aq+b+cq^{-1}=0 \tag{d-2}$$
$$aq+b+cq^*/|q|^2=0 \tag{d-3}$$
$$qc^*/|q|^2+b^*+q^*a^*=0 \tag{d-4}$$
$$aqa^*+ba^*+cq^*a^*/|q|^2=0 \tag{d-5}$$
$$cqc^*/|q|^2+cb^*+cq^*a^*=0 \tag{d-6}$$
$$|q|^2aqa^*+|q|^2ba^*+cq^*a^*=0 \tag{d-7}$$
$$|q|^2aqa^*+|q|^2ba^*-cqc^*/|q|^2-cb^*=0 \tag{d-8}$$
$$|q|^4aqa^*+|q|^4ba^*-cqc^*-|q|^2cb^*=0 \tag{d-9}$$
$$|q|^4aqa^*-cqc^*=|q|^2cb^*-|q|^4ba^* \tag{d-10}$$

Let, $q=\lambda\cdot p\,,\qquad \lambda\in\mathbb{R},\ |p|=1$ (d-11)

$$\underline{\lambda^5apa^*-\lambda cpc^*=\lambda^2cb^*-\lambda^4ba^*} \tag{d-12}$$

Now follow the dot marks " · " to see the changes;

$$\lambda^5apa^*\cdot c-\lambda cpc^*\cdot c=\lambda^2cb^*\cdot c-\lambda^4ba^*\cdot c \tag{d-13}$$
$$\lambda^5apa^*c-\lambda cp\cdot c^*c=\lambda^2cb^*c-\lambda^4ba^*c \tag{d-14}$$
$$\lambda^5apa^*c-\lambda cc^*\cdot cp=\lambda^2cb^*c-\lambda^4ba^*c \tag{d-15}$$

$$\lambda^5apa^*\cdot a-\lambda cpc^*\cdot a=\lambda^2cb^*\cdot a-\lambda^4ba^*\cdot a \tag{d-16}$$
$$\lambda^5ap\cdot a^*a-\lambda cpc^*a=\lambda^2cb^*a-\lambda^4ba^*a \tag{d-17}$$
$$\lambda^5aa^*\cdot ap-\lambda cpc^*a=\lambda^2cb^*a-\lambda^4ba^*a \tag{d-18}$$

$$\lambda^5a^*\cdot apa^*c-\lambda a^*\cdot cc^*cp=\lambda^2a^*\cdot cb^*c-\lambda^4a^*\cdot ba^*c \tag{d-19}$$
$$\lambda^5pa^*c\cdot a^*a-\lambda a^*cc^*cp=\lambda^2a^*cb^*c-\lambda^4a^*ba^*c \tag{d-20}$$
$$\lambda^5c\cdot pa^*ca^*a-\lambda c\cdot a^*cc^*cp=\lambda^2c\cdot a^*cb^*c-\lambda^4c\cdot a^*ba^*c \tag{d-21}$$

$$\lambda^5cpa^*c\cdot a^*a-\lambda ca^*cc^*cp=\lambda^2ca^*cb^*c-\lambda^4ca^*ba^*c \tag{d-22}$$
$$\lambda^5a^*a\cdot cpa^*c-\lambda ca^*cc^*cp=\lambda^2ca^*cb^*c-\lambda^4ca^*ba^*c \tag{d-23}$$

$$\lambda^5a^*a\cdot aa^*ap-\lambda a^*a\cdot cpc^*a=\lambda^2a^*a\cdot cb^*a-\lambda^4a^*a\cdot ba^*a \tag{d-24}$$
$$\lambda^5a^*aaa^*ap-\lambda a^*acpc^*a=\lambda^2a^*acb^*a-\lambda^4a^*aba^*a \tag{d-25}$$

$$\lambda^5a^*acpa^*c-\lambda ca^*cc^*cp=\lambda^2ca^*cb^*c-\lambda^4ca^*ba^*c \tag{d-26}$$
$$\lambda^4\cdot\lambda^5a^*aaa^*ap-\lambda^4\cdot\lambda a^*acpc^*a=\lambda^4\cdot\lambda^2a^*acb^*a-\lambda^4\cdot\lambda^4a^*aba^*a \tag{d-27}$$
$$\lambda^9a^*aaa^*ap-\lambda^5a^*acpc^*a=\lambda^6a^*acb^*a-\lambda^8a^*aba^*a \tag{d-28}$$

$$\lambda^9a^*aaa^*ap+\lambda ca^*cc^*cp-\lambda^5a^*acp\cdot(a^*c+c^*a)=\lambda^6a^*acb^*a-\lambda^8a^*aba^*a-\lambda^2ca^*cb^*c+\lambda^4ca^*ba^*c \tag{d-29}$$
$$\lambda^9a^*aaa^*ap+\lambda ca^*cc^*cp-\lambda^5a^*ac\cdot(a^*c+c^*a)\cdot p=\lambda^6|a|^2(cb^*-\lambda^2ba^*)a-\lambda^2ca^*(cb^*-\lambda^2ba^*)c \tag{d-30}$$

$$\underline{\left(\lambda^8|a|^4a+|c|^2ca^*c-\lambda^4a^*ac(a^*c+c^*a)\right)\cdot p=\lambda^5|a|^2(cb^*-\lambda^2ba^*)a-\lambda ca^*(cb^*-\lambda^2ba^*)c} \tag{d-31}$$

Next, multiply both sides by, a^*,

$$a^* \cdot \left(\lambda^8|a|^4 a + |c|^2 c a^* c - \lambda^4 a^* a c (a^* c + c^* a)\right) \cdot p = \lambda^5 |a|^2 a^* \cdot (cb^* - \lambda^2 b a^*) a - \lambda a^* \cdot c a^* (cb^* - \lambda^2 b a^*) c \tag{d-32}$$

Working out the L.H.S factor,

$$\begin{aligned}
& a^* \cdot \left(\lambda^8|a|^4 a + |c|^2 c a^* c - \lambda^4 a^* a c (a^* c + c^* a)\right) && \text{(d-33)}\\
&= \left(\lambda^8|a|^4 a^* a + |c|^2 a^* c a^* c - \lambda^4 a^* a a^* c (a^* c + c^* a)\right) && \text{(d-34)}\\
&= \left(\lambda^8|a|^6 + |c|^2 (a^* c)^2 - \lambda^4 |a|^2 (a^* c a^* c + a^* c c^* a)\right) && \text{(d-35)}\\
&= \left(\lambda^8|a|^6 + |c|^2 (a^* c)^2 - \lambda^4 |a|^2 ((a^* c)^2 + |a|^2 |c|^2)\right) && \text{(d-36)}\\
&= \left(\lambda^4|a|^2 (\lambda^4|a|^4 - (a^* c)^2) - |c|^2 (\lambda^4|a|^4 - (a^* c)^2)\right) && \text{(d-37)}
\end{aligned}$$

Hence, we arrive at the following,

$$\left(\lambda^4|a|^2 - |c|^2\right) \cdot \left(\lambda^4|a|^4 - (a^* c)^2\right) \cdot p = \lambda^5 |a|^2 a^* (cb^* - \lambda^2 b a^*) a - \lambda a^* c a^* (cb^* - \lambda^2 b a^*) c \tag{d-38}$$

We can now multiply both sides by the conjugate factor, $(\lambda^4|a|^2 - (a^* c)^2)^*$, to reduce the L.H.S. factor to scalar, and then rearrange the whole equation to have, p, by itself on the left, obtaining,

$$p \;=\; \frac{(\lambda^4|a|^2 - (a^* c)^2)^* \cdot (\lambda^5 |a|^2 a^* (cb^* - \lambda^2 b a^*) a - \lambda a^* c a^* (cb^* - \lambda^2 b a^*) c)}{(\lambda^4|a|^2 - |c|^2) \cdot (\lambda^4|a|^2 - (a^* c)^2)^* \cdot (\lambda^4|a|^4 - (a^* c)^2)} \tag{d-39}$$

Rearrange the numerator;

$$\begin{aligned}
& \left(\lambda^4|a|^4 - (a^* c)^2\right)^* \cdot \left(\lambda^5 |a|^2 a^* (cb^* - \lambda^2 b a^*) a - \lambda a^* c a^* (cb^* - \lambda^2 b a^*) c\right) && \text{(d-40)}\\
&= \left(\lambda^4|a|^4 - (c^* a)^2\right) \cdot \left(\lambda^5 |a|^2 a^* (cb^* - \lambda^2 b a^*) a - \lambda a^* c a^* (cb^* - \lambda^2 b a^*) c\right) && \text{(d-41)}\\
&= \lambda^5 |a|^2 \cdot (\lambda^4|a|^4 - (c^* a)^2) \cdot a^* (cb^* - \lambda^2 b a^*) a - \lambda \cdot (\lambda^4|a|^4 - (c^* a)^2) \cdot a^* c a^* (cb^* - \lambda^2 b a^*) c && \text{(d-42)}\\
&= \lambda^5 |a|^2 \cdot (\lambda^4|a|^4 - c^* a c^* a) \cdot a^* (cb^* - \lambda^2 b a^*) a - \lambda \cdot (\lambda^4|a|^4 - c^* a c^* a) \cdot a^* c a^* (cb^* - \lambda^2 b a^*) c && \text{(d-43)}\\
&= \lambda^5 |a|^2 \cdot (\lambda^4|a|^4 a^* - c^* a c^* a a^*) \cdot (cb^* - \lambda^2 b a^*) a - \lambda \cdot (\lambda^4|a|^4 a^* c a^* - c^* a c^* a a^* c a^*) \cdot (cb^* - \lambda^2 b a^*) c && \text{(d-44)}\\
&= \lambda^5 |a|^4 \cdot (\lambda^4|a|^2 a^* - c^* a c^*) \cdot (cb^* - \lambda^2 b a^*) a - \lambda |a|^4 \cdot (\lambda^4 a^* c a^* - |c|^2 c^*) \cdot (cb^* - \lambda^2 b a^*) c && \text{(d-45)}
\end{aligned}$$

$$= |a|^4 \cdot \left(\, \lambda^5 \cdot (\lambda^4|a|^2 a^* - c^* a c^*) \cdot (cb^* - \lambda^2 b a^*) a + \lambda \cdot (|c|^2 c^* - \lambda^4 a^* c a^*) \cdot (cb^* - \lambda^2 b a^*) c \,\right) \tag{d-46}$$

Rearrange the denominator's conjugate factor product;

$$\begin{aligned}
& \left(\lambda^4|a|^4 - (a^* c)^2\right)^* \cdot \left(\lambda^4|a|^4 - (a^* c)^2\right) && \text{(d-47)}\\
&= \left(\lambda^4|a|^4 - ((a^* c)^*)^2\right) \cdot \left(\lambda^4|a|^4 - (a^* c)^2\right) && \text{(d-48)}\\
&= \left(\lambda^2|a|^2 - (a^* c)^*\right) \left(\lambda^2|a|^2 + (a^* c)^*\right) \cdot \left(\lambda^2|a|^2 - (a^* c)\right) \left(\lambda^2|a|^2 + (a^* c)\right) && \text{(d-49)}\\
&= \left(\lambda^2|a|^2 - (a^* c)^*\right) \left(\lambda^2|a|^2 - (a^* c)\right) \cdot \left(\lambda^2|a|^2 + (a^* c)^*\right) \left(\lambda^2|a|^2 + (a^* c)\right) && \text{(d-50)}\\
&= \left|\lambda^2|a|^2 - (a^* c)^*\right|^2 \cdot \left|\lambda^2|a|^2 + (a^* c)^*\right|^2 && \text{(d-51)}\\
&= \left|\lambda^2|a|^2 - (a^* c)\right|^2 \cdot \left|\lambda^2|a|^2 + (a^* c)\right|^2 && \text{(d-52)}\\
&= \left|\lambda^2 a^* a - a^* c\right|^2 \cdot \left|\lambda^2 a^* a + a^* c\right|^2 && \text{(d-53)}\\
&= |a^*|^2 \left|\lambda^2 a - c\right|^2 \cdot |a^*|^2 \left|\lambda^2 a + c\right|^2 && \text{(d-54)}\\
&= |a|^4 \cdot \left|\lambda^2 a - c\right|^2 \cdot \left|\lambda^2 a + c\right|^2 && \text{(d-55)}
\end{aligned}$$

After cancelling out the common $|a|^4$ factor, we obtain the usual formula for p, whence, $q = \lambda \cdot p$, so;

$$q \;=\; \lambda \cdot \frac{\left(\, \lambda^5 \cdot (\lambda^4|a|^2 a^* - c^* a c^*) \cdot (cb^* - \lambda^2 b a^*) a + \lambda \cdot (|c|^2 c^* - \lambda^4 a^* c a^*) \cdot (cb^* - \lambda^2 b a^*) c \,\right)}{(\lambda^4|a|^2 - |c|^2) \cdot \left|\lambda^2 a - c\right|^2 \cdot \left|\lambda^2 a + c\right|^2} \tag{d-56}$$

Q.E.D.

$$q = \lambda \cdot \frac{(\lambda^6|a|^2a^* - \lambda^2c^*ac^*)(\lambda^3cb^*a - \lambda^5ba^*a) + (|c|^2c^* - \lambda^4a^*ca^*)(\lambda cb^*c - \lambda^3ba^*c)}{(\lambda^4|a|^2 - |c|^2)((\lambda^4|a|^2 + |c|^2)^2 - \lambda^4((ac^*) + (ac^*)^*)^2)} \tag{d-57}$$

$$= Tq \cdot Uq$$

$\underline{q^2a + qb + c = 0}$: Consider...Quaternion Quadratic Equation #2:

$$q = \lambda \cdot \frac{(\lambda^6|a|^2a - \lambda^2ca^*c)(\lambda^3b^*ca^* - \lambda^5a^*ba^*) + (|c|^2c - \lambda^4ac^*a)(\lambda b^*cc^* - \lambda^3a^*bc^*)}{(\lambda^4|a|^2 - |c|^2)((\lambda^4|a|^2 + |c|^2)^2 - \lambda^4(ac^* + (ac^*)^*)^2)} \tag{d-58}$$

$$= Tq \cdot Uq \tag{d-59}$$

$\underline{qaq + bq + c = 0}$: Consider...Quaternion Quadratic Equation #3:

$$q = \lambda \cdot \frac{(\lambda^3cab^* - \lambda^5aa^*b)(\lambda^6a^*aa^* - \lambda^2cac) + (\lambda c^*cb^* - \lambda^3a^*c^*b)(cc^*c - \lambda^4a^*c^*a^*)}{(\lambda^4|a|^2 - |c|^2) \cdot ((\lambda^4|a|^2 + |c|^2)^2 - \lambda^4((ca) + (ca)^*)^2)} \tag{d-60}$$

$$= Tq \cdot Uq \tag{d-61}$$

APPENDIX E: Quaternion Formula Component Expansions

$$u_1 = u10 + u11 \cdot i + u12 \cdot j + u13 \cdot k \tag{e-1}$$
$$u_2 = u20 + u21 \cdot i + u22 \cdot j + u23 \cdot k \tag{e-2}$$

$$\checkmark\ s = u_1 + u_2 = \tag{e-3}$$
$$+(u10 + u20) \tag{e-4}$$
$$+(u11 + u21) \cdot i \tag{e-5}$$
$$+(u12 + u22) \cdot j \tag{e-6}$$
$$+(u13 + u23) \cdot k \tag{e-7}$$

$$\checkmark\ d = u_1 - u_2 = \tag{e-8}$$
$$+(u10 - u20) \tag{e-9}$$
$$+(u11 - u21) \cdot i \tag{e-10}$$
$$+(u12 - u22) \cdot j \tag{e-11}$$
$$+(u13 - u23) \cdot k \tag{e-12}$$

$$\checkmark\ s^* = u_1^* + u_2^* = \tag{e-13}$$
$$+(u10 + u20) \tag{e-14}$$
$$-(u11 + u21) \cdot i \tag{e-15}$$
$$-(u12 + u22) \cdot j \tag{e-16}$$
$$-(u13 + u23) \cdot k \tag{e-17}$$

$$\checkmark\ d^* = u_1^* - u_2^* = \tag{e-18}$$
$$+(u10 - u20) \tag{e-19}$$
$$-(u11 - u21) \cdot i \tag{e-20}$$
$$-(u12 - u22) \cdot j \tag{e-21}$$
$$-(u13 - u23) \cdot k \tag{e-22}$$

$$ds = (u_1 - u_2)(u_1 + u_2) = u_1^2 - u_2^2 - u_2u_1 + u_1u_2 = \tag{e-23}$$
$$+1.u10.u10 - 1.u11.u11 - 1.u12.u12 - 1.u13.u13 - 1.u20.u20 + 1.u21.u21 + 1.u22.u22 + 1.u23.u23 \tag{e-24}$$
$$+(+2.u10.u11 + 2.u12.u23 - 2.u13.u22 - 2.u20.u21) \cdot i \tag{e-25}$$
$$+(+2.u10.u12 - 2.u11.u23 + 2.u13.u21 - 2.u20.u22) \cdot j \tag{e-26}$$
$$+(+2.u10.u13 + 2.u11.u22 - 2.u12.u21 - 2.u20.u23) \cdot k \tag{e-27}$$

$$sd = (u_1 + u_2)(u_1 - u_2) = u_1^2 - u_2^2 + u_2u_1 - u_1u_2 = \tag{e-28}$$
$$+1.u10.u10 - 1.u11.u11 - 1.u12.u12 - 1.u13.u13 - 1.u20.u20 + 1.u21.u21 + 1.u22.u22 + 1.u23.u23 \tag{e-29}$$
$$+(+2.u10.u11 - 2.u12.u23 + 2.u13.u22 - 2.u20.u21) \cdot i \tag{e-30}$$
$$+(+2.u10.u12 + 2.u11.u23 - 2.u13.u21 - 2.u20.u22) \cdot j \tag{e-31}$$
$$+(+2.u10.u13 - 2.u11.u22 + 2.u12.u21 - 2.u20.u23) \cdot k \tag{e-32}$$

$$\{d, s\} = ds + sd = 2(u_1^2 - u_2^2) = \tag{e-33}$$
$$+2.u10.u10 - 2.u11.u11 - 2.u12.u12 - 2.u13.u13 - 2.u20.u20 + 2.u21.u21 + 2.u22.u22 + 2.u23.u23 \tag{e-34}$$
$$+(+4.u10.u11 - 4.u20.u21) \cdot i \tag{e-35}$$
$$+(+4.u10.u12 - 4.u20.u22) \cdot j \tag{e-36}$$
$$+(+4.u10.u13 - 4.u20.u23) \cdot k \tag{e-37}$$

$$[d, s] = ds - sd = 2(u_1u_2 - u_2u_1) = \tag{e-38}$$
$$+0 \tag{e-39}$$
$$+(+4.u12.u23 - 4.u13.u22) \cdot i \tag{e-40}$$
$$+(-4.u11.u23 + 4.u13.u21) \cdot j \tag{e-41}$$
$$+(+4.u11.u22 - 4.u12.u21) \cdot k \tag{e-42}$$

$$\{d, s\}^* = (ds + sd)^* = s^*d^* + d^*s^* = 2({u_1^*}^2 - {u_2^*}^2) = \tag{e-43}$$
$$+2.u10.u10 - 2.u11.u11 - 2.u12.u12 - 2.u13.u13 - 2.u20.u20 + 2.u21.u21 + 2.u22.u22 + 2.u23.u23 \tag{e-44}$$
$$+(-4.u10.u11 + 4.u20.u21) \cdot i \tag{e-45}$$
$$+(-4.u10.u12 + 4.u20.u22) \cdot j \tag{e-46}$$
$$+(-4.u10.u13 + 4.u20.u23) \cdot k \tag{e-47}$$

$$[d, s]^* = (ds - sd)^* = s^*d^* - d^*s^* = 2(u_2^*u_1^* - u_1^*u_2^*) = \tag{e-48}$$
$$+0 \tag{e-49}$$
$$+(-4.u12.u23 + 4.u13.u22) \cdot i \tag{e-50}$$
$$+(+4.u11.u23 - 4.u13.u21) \cdot j \tag{e-51}$$
$$+(-4.u11.u22 + 4.u12.u21) \cdot k \tag{e-52}$$

$$d^2 = dd = (u_1 - u_2)(u_1 - u_2) = u_1^2 + u_2^2 - u_2 u_1 - u_1 u_2 = \tag{e-53}$$
$$+1.u10.u10 - 2.u10.u20 - 1.u11.u11 + 2.u11.u21 - 1.u12.u12 + 2.u12.u22 \tag{e-54}$$
$$-1.u13.u13 + 2.u13.u23 + 1.u20.u20 - 1.u21.u21 - 1.u22.u22 - 1.u23.u23 \tag{e-55}$$
$$+(+2.u10.u11 - 2.u10.u21 - 2.u11.u20 + 2.u20.u21) \cdot i \tag{e-56}$$
$$+(+2.u10.u12 - 2.u10.u22 - 2.u12.u20 + 2.u20.u22) \cdot j \tag{e-57}$$
$$+(+2.u10.u13 - 2.u10.u23 - 2.u13.u20 + 2.u20.u23) \cdot k \tag{e-58}$$

$$(d^*)^2 = d^* d^* = (u_1^* - u_2^*)(u_1^* - u_2^*) = {u_1^*}^2 + {u_2^*}^2 - u_1^* u_2^* - u_2^* u_1^* = \tag{e-59}$$
$$+1.u10.u10 - 2.u10.u20 - 1.u11.u11 + 2.u11.u21 - 1.u12.u12 + 2.u12.u22 \tag{e-60}$$
$$-1.u13.u13 + 2.u13.u23 + 1.u20.u20 - 1.u21.u21 - 1.u22.u22 - 1.u23.u23 \tag{e-61}$$
$$+(-2.u10.u11 + 2.u10.u21 + 2.u11.u20 - 2.u20.u21) \cdot i \tag{e-62}$$
$$+(-2.u10.u12 + 2.u10.u22 + 2.u12.u20 - 2.u20.u22) \cdot j \tag{e-63}$$
$$+(-2.u10.u13 + 2.u10.u23 + 2.u13.u20 - 2.u20.u23) \cdot k \tag{e-64}$$

$$|d|^2 = dd^* = d^* d = (u_1 - u_2)(u_1^* - u_2^*) = 2 - u_2 u_1^* - u_1 u_2^* = \tag{e-65}$$
$$+1.u10.u10 - 2.u10.u20 + 1.u11.u11 - 2.u11.u21 + 1.u12.u12 - 2.u12.u22 \tag{e-66}$$
$$+1.u13.u13 - 2.u13.u23 + 1.u20.u20 + 1.u21.u21 + 1.u22.u22 + 1.u23.u23 \tag{e-67}$$
$$+0 \cdot i \tag{e-68}$$
$$+0 \cdot j \tag{e-69}$$
$$+0 \cdot k \tag{e-70}$$

$$[d, s] \cdot d^* = (ds - sd)d^* = dsd^* - sdd^* = dsd^* - dd^* s = d(sd^* - d^* s) = d[s, d^*] = -d[d^*, s] = \tag{e-71}$$
$$+0 \tag{e-72}$$
$$+[+4.u10.u12.u23 - 4.u10.u13.u22 + 4.u11.u12.u22 + 4.u11.u13.u23 - 4.u11.u22.u22 - 4.u11.u23.u23 \tag{e-73}$$
$$-4.u12.u12.u21 - 4.u12.u20.u23 + 4.u12.u21.u22 - 4.u13.u13.u21 + 4.u13.u20.u22 + 4.u13.u21.u23\ ] \cdot i \tag{e-74}$$
$$+[-4.u10.u11.u23 + 4.u10.u13.u21 - 4.u11.u11.u22 + 4.u11.u12.u21 + 4.u11.u20.u23 + 4.u11.u21.u22 \tag{e-75}$$
$$+4.u12.u13.u23 - 4.u12.u21.u21 - 4.u12.u23.u23 - 4.u13.u13.u22 - 4.u13.u20.u21 + 4.u13.u22.u23\ ] \cdot j \tag{e-76}$$
$$+[+4.u10.u11.u22 - 4.u10.u12.u21 - 4.u11.u11.u23 + 4.u11.u13.u21 - 4.u11.u20.u22 + 4.u11.u21.u23 \tag{e-77}$$
$$-4.u12.u12.u23 + 4.u12.u13.u22 + 4.u12.u20.u21 + 4.u12.u22.u23 - 4.u13.u21.u21 - 4.u13.u22.u22\ ] \cdot k \tag{e-78}$$

$$d \cdot [d, s]^* = d(ds - sd)^* = d(s^* d^* - d^* s^*) = ds^* d^* - dd^* s^* = ds^* d^* - s^* dd^* = (ds^* - s^* d)d^* = [d, s^*]d^* = \tag{e-79}$$
$$+0 \tag{e-80}$$
$$+[-4.u10.u12.u23 + 4.u10.u13.u22 - 4.u11.u12.u22 - 4.u11.u13.u23 + 4.u11.u22.u22 + 4.u11.u23.u23 \tag{e-81}$$
$$+4.u12.u12.u21 + 4.u12.u20.u23 - 4.u12.u21.u22 + 4.u13.u13.u21 - 4.u13.u20.u22 - 4.u13.u21.u23\ ] \cdot i \tag{e-82}$$
$$+[+4.u10.u11.u23 - 4.u10.u13.u21 + 4.u11.u11.u22 - 4.u11.u12.u21 - 4.u11.u20.u23 - 4.u11.u21.u22 \tag{e-83}$$
$$-4.u12.u13.u23 + 4.u12.u21.u21 + 4.u12.u23.u23 + 4.u13.u13.u22 + 4.u13.u20.u21 - 4.u13.u22.u23\ ] \cdot j \tag{e-84}$$
$$+[-4.u10.u11.u22 + 4.u10.u12.u21 + 4.u11.u11.u23 - 4.u11.u13.u21 + 4.u11.u20.u22 - 4.u11.u21.u23 \tag{e-85}$$
$$+4.u12.u12.u23 - 4.u12.u13.u22 - 4.u12.u20.u21 - 4.u12.u22.u23 + 4.u13.u21.u21 + 4.u13.u22.u22\ ] \cdot k \tag{e-86}$$

$$d^2 - (d^*)^2 = \tag{e-87}$$
$$+0 \tag{e-88}$$
$$+[+4.u10.u11 - 4.u10.u21 - 4.u11.u20 + 4.u20.u21\ ] \cdot i \tag{e-89}$$
$$+[+4.u10.u12 - 4.u10.u22 - 4.u12.u20 + 4.u20.u22\ ] \cdot j \tag{e-90}$$
$$+[+4.u10.u13 - 4.u10.u23 - 4.u13.u20 + 4.u20.u23\ ] \cdot k \tag{e-91}$$

$$|d^2 - (d^*)^2|^2 = \tag{e-92}$$
$$+16.u10.u10.u11.u11 - 32.u10.u10.u11.u21 + 16.u10.u10.u12.u12 - 32.u10.u10.u12.u22 + 16.u10.u10.u13.u13 \tag{e-93}$$
$$-32.u10.u10.u13.u23 + 16.u10.u10.u21.u21 + 16.u10.u10.u22.u22 + 16.u10.u10.u23.u23 - 32.u10.u11.u11.u20 \tag{e-94}$$
$$+64.u10.u11.u20.u21 - 32.u10.u12.u12.u20 + 64.u10.u12.u20.u22 - 32.u10.u13.u13.u20 + 64.u10.u13.u20.u23 \tag{e-95}$$
$$-32.u10.u20.u21.u21 - 32.u10.u20.u22.u22 - 32.u10.u20.u23.u23 + 16.u11.u11.u20.u20 - 32.u11.u20.u20.u21 \tag{e-96}$$
$$+16.u12.u12.u20.u20 - 32.u12.u20.u20.u22 + 16.u13.u13.u20.u20 - 32.u13.u20.u20.u23 + 16.u20.u20.u21.u21 \tag{e-97}$$
$$+16.u20.u20.u22.u22 + 16.u20.u20.u23.u23 \tag{e-98}$$
$$+0 \cdot i \tag{e-99}$$
$$+0 \cdot j \tag{e-100}$$
$$+0 \cdot k \tag{e-101}$$

$$\{d,s\} = ds + sd = \tag{e-102}$$
$$+2.u10.u10 - 2.u11.u11 - 2.u12.u12 - 2.u13.u13 - 2.u20.u20 + 2.u21.u21 + 2.u22.u22 + 2.u23.u23 \tag{e-103}$$
$$+[+4.u10.u11 - 4.u20.u21\ ] \cdot i \tag{e-104}$$
$$+[+4.u10.u12 - 4.u20.u22\ ] \cdot j \tag{e-105}$$
$$+[+4.u10.u13 - 4.u20.u23\ ] \cdot k \tag{e-106}$$

$$\{d,s\}^* = (ds + sd)^* = s^*d^* + d^*s^* = \tag{e-107}$$
$$+2.u10.u10 - 2.u11.u11 - 2.u12.u12 - 2.u13.u13 - 2.u20.u20 + 2.u21.u21 + 2.u22.u22 + 2.u23.u23 \tag{e-108}$$
$$+[-4.u10.u11 + 4.u20.u21\ ] \cdot i \tag{e-109}$$
$$+[-4.u10.u12 + 4.u20.u22\ ] \cdot j \tag{e-110}$$
$$+[-4.u10.u13 + 4.u20.u23\ ] \cdot k \tag{e-111}$$

$$[\{d,s\},d\,] = \tag{e-112}$$
$$+0 \tag{e-113}$$
$$+[-8.u10.u12.u23 + 8.u10.u13.u22 + 8.u12.u20.u23 - 8.u13.u20.u22\ ] \cdot i \tag{e-114}$$
$$+[+8.u10.u11.u23 - 8.u10.u13.u21 - 8.u11.u20.u23 + 8.u13.u20.u21\ ] \cdot j \tag{e-115}$$
$$+[-8.u10.u11.u22 + 8.u10.u12.u21 + 8.u11.u20.u22 - 8.u12.u20.u21\ ] \cdot k \tag{e-116}$$

$$[\{d,s\}^*,d\,] = \tag{e-117}$$
$$+0 \tag{e-118}$$
$$+[+8.u10.u12.u23 - 8.u10.u13.u22 - 8.u12.u20.u23 + 8.u13.u20.u22\ ] \cdot i \tag{e-119}$$
$$+[-8.u10.u11.u23 + 8.u10.u13.u21 + 8.u11.u20.u23 - 8.u13.u20.u21\ ] \cdot j \tag{e-120}$$
$$+[+8.u10.u11.u22 - 8.u10.u12.u21 - 8.u11.u20.u22 + 8.u12.u20.u21\ ] \cdot k \tag{e-121}$$

$$\{d,[\{d,s\},d\,]\} = \tag{e-122}$$
$$+0 \tag{e-123}$$
$$+[-16.u10.u10.u12.u23 + 16.u10.u10.u13.u22 + 32.u10.u12.u20.u23 - 32.u10.u13.u20.u22 \tag{e-124}$$
$$\quad -16.u12.u20.u20.u23 + 16.u13.u20.u20.u22\ ] \cdot i \tag{e-125}$$
$$+[+16.u10.u10.u11.u23 - 16.u10.u10.u13.u21 - 32.u10.u11.u20.u23 + 32.u10.u13.u20.u21 \tag{e-126}$$
$$\quad +16.u11.u20.u20.u23 - 16.u13.u20.u20.u21\ ] \cdot j \tag{e-127}$$
$$+[-16.u10.u10.u11.u22 + 16.u10.u10.u12.u21 + 32.u10.u11.u20.u22 - 32.u10.u12.u20.u21 \tag{e-128}$$
$$\quad -16.u11.u20.u20.u22 + 16.u12.u20.u20.u21\ ] \cdot k \tag{e-129}$$

$$\{d,[\{d,s\}^*,d\,]\} = \tag{e-130}$$
$$+0 \tag{e-131}$$
$$+[+16.u10.u10.u12.u23 - 16.u10.u10.u13.u22 - 32.u10.u12.u20.u23 + 32.u10.u13.u20.u22 \tag{e-132}$$
$$\quad +16.u12.u20.u20.u23 - 16.u13.u20.u20.u22\ ] \cdot i \tag{e-133}$$
$$+[-16.u10.u10.u11.u23 + 16.u10.u10.u13.u21 + 32.u10.u11.u20.u23 - 32.u10.u13.u20.u21 \tag{e-134}$$
$$\quad -16.u11.u20.u20.u23 + 16.u13.u20.u20.u21\ ] \cdot j \tag{e-135}$$
$$+[+16.u10.u10.u11.u22 - 16.u10.u10.u12.u21 - 32.u10.u11.u20.u22 + 32.u10.u12.u20.u21 \tag{e-136}$$
$$\quad +16.u11.u20.u20.u22 - 16.u12.u20.u20.u21\ ] \cdot k \tag{e-137}$$

$$\{d,[\{d,s\},d\,]\} + \{d,[\{d,s\}^*,d\,]\} = \qquad \cdot * = \texttt{identity} = * \cdot \tag{e-138}$$
$$+0 \tag{e-139}$$
$$+0 \cdot i \tag{e-140}$$
$$+0 \cdot j \tag{e-141}$$
$$+0 \cdot k \tag{e-142}$$

$$\boldsymbol{nT1} = [\ 5d^2 \cdot d \cdot [d,s]^* - 3|d|^2 \cdot [d,s] \cdot d^*\ ] \cdot [\ (d^2 - (d^*)^2)((d^*)^2 - d^2)\ ] = \tag{e-143}$$
$$+0 \tag{e-144}$$
$$+[-512.u10.u10.u10.u10.u10.u11.u11.u12.u23 + 512.u10.u10.u10.u10.u10.u11.u11.u13.u22 + \cdots\ ] \cdot i \tag{e-145}$$
$$+[+512.u10.u10.u10.u10.u10.u11.u11.u11.u23 - 512.u10.u10.u10.u10.u10.u11.u11.u13.u21 - \cdots\ ] \cdot j \tag{e-146}$$
$$+[-512.u10.u10.u10.u10.u10.u11.u11.u11.u22 + 512.u10.u10.u10.u10.u10.u11.u11.u12.u21 + \cdots\ ] \cdot k \tag{e-147}$$

$$\boldsymbol{nT2} = (-5d^2 + (d^*)^2) \cdot d^2 \cdot d \cdot \{d,[\{d,s\}^*,d\,]\} + (-7d^2 + 3(d^*)^2) \cdot |d|^2 \cdot \{d,[\{d,s\},d\,]\} \cdot d^* = \tag{e-148}$$
$$+0 \tag{e-149}$$
$$+[+512.u10.u10.u10.u10.u10.u11.u11.u12.u23 - 512.u10.u10.u10.u10.u10.u11.u11.u13.u22 - \cdots\ ] \cdot i \tag{e-150}$$
$$+[-512.u10.u10.u10.u10.u10.u11.u11.u11.u23 + 512.u10.u10.u10.u10.u10.u11.u11.u13.u21 + \cdots\ ] \cdot j \tag{e-151}$$
$$+[+512.u10.u10.u10.u10.u10.u11.u11.u11.u22 - 512.u10.u10.u10.u10.u10.u11.u11.u12.u21 - \cdots\ ] \cdot k \tag{e-152}$$

$$\boldsymbol{nT1 + nT2 \equiv 0} \qquad \cdot * = \texttt{identity} = * \cdot \qquad narh(\lambda = m) = 0 \quad \texttt{confirmed!} \tag{e-153}$$

$$\{d,[\{d,s\},d\,]\}\cdot d^{*}= \tag{e-154}$$
$$+0 \tag{e-155}$$
$$+[-16.u10.u10.u10.u12.u23+16.u10.u10.u10.u13.u22-16.u10.u10.u11.u12.u22-16.u10.u10.u11.u13.u23 \tag{e-156}$$
$$+16.u10.u10.u11.u22.u22+16.u10.u10.u11.u23.u23+16.u10.u10.u12.u12.u21+48.u10.u10.u12.u20.u23 \tag{e-157}$$
$$-16.u10.u10.u12.u21.u22+16.u10.u10.u13.u13.u21-48.u10.u10.u13.u20.u22-16.u10.u10.u13.u21.u23 \tag{e-158}$$
$$+32.u10.u11.u12.u20.u22+32.u10.u11.u13.u20.u23-32.u10.u11.u20.u22.u22-32.u10.u11.u20.u23.u23 \tag{e-159}$$
$$-32.u10.u12.u12.u20.u21-48.u10.u12.u20.u20.u23+32.u10.u12.u20.u21.u22-32.u10.u13.u13.u20.u21 \tag{e-160}$$
$$+48.u10.u13.u20.u20.u22+32.u10.u13.u20.u21.u23-16.u11.u12.u20.u20.u22-16.u11.u13.u20.u20.u23 \tag{e-161}$$
$$+16.u11.u20.u20.u22.u22+16.u11.u20.u20.u23.u23+16.u12.u12.u20.u20.u21+16.u12.u20.u20.u20.u23 \tag{e-162}$$
$$-16.u12.u20.u20.u21.u22+16.u13.u13.u20.u20.u21-16.u13.u20.u20.u20.u22-16.u13.u20.u20.u21.u23\,]\cdot i \tag{e-163}$$
$$+[+16.u10.u10.u10.u11.u23-16.u10.u10.u10.u13.u21+16.u10.u10.u11.u11.u22-16.u10.u10.u11.u12.u21 \tag{e-164}$$
$$-48.u10.u10.u11.u20.u23-16.u10.u10.u11.u21.u22-16.u10.u10.u12.u13.u23+16.u10.u10.u12.u21.u21 \tag{e-165}$$
$$+16.u10.u10.u12.u23.u23+16.u10.u10.u13.u13.u22+48.u10.u10.u13.u20.u21-16.u10.u10.u13.u22.u23 \tag{e-166}$$
$$-32.u10.u11.u11.u20.u22+32.u10.u11.u12.u20.u21+48.u10.u11.u20.u20.u23+32.u10.u11.u20.u21.u22 \tag{e-167}$$
$$+32.u10.u12.u13.u20.u23-32.u10.u12.u20.u21.u21-32.u10.u12.u20.u23.u23-32.u10.u13.u13.u20.u22 \tag{e-168}$$
$$-48.u10.u13.u20.u20.u21+32.u10.u13.u20.u22.u23+16.u11.u11.u20.u20.u22-16.u11.u12.u20.u20.u21 \tag{e-169}$$
$$-16.u11.u20.u20.u20.u23-16.u11.u20.u20.u21.u22-16.u12.u13.u20.u20.u23+16.u12.u20.u20.u21.u21 \tag{e-170}$$
$$+16.u12.u20.u20.u23.u23+16.u13.u13.u20.u20.u22+16.u13.u20.u20.u20.u21-16.u13.u20.u20.u22.u23\,]\cdot j \tag{e-171}$$
$$+[-16.u10.u10.u10.u11.u22+16.u10.u10.u10.u12.u21+16.u10.u10.u11.u11.u23-16.u10.u10.u11.u13.u21 \tag{e-172}$$
$$+48.u10.u10.u11.u20.u22-16.u10.u10.u11.u21.u23+16.u10.u10.u12.u12.u23-16.u10.u10.u12.u13.u22 \tag{e-173}$$
$$-48.u10.u10.u12.u20.u21-16.u10.u10.u12.u22.u23+16.u10.u10.u13.u21.u21+16.u10.u10.u13.u22.u22 \tag{e-174}$$
$$-32.u10.u11.u11.u20.u23+32.u10.u11.u13.u20.u21-48.u10.u11.u20.u20.u22+32.u10.u11.u20.u21.u23 \tag{e-175}$$
$$-32.u10.u12.u12.u20.u23+32.u10.u12.u13.u20.u22+48.u10.u12.u20.u20.u21+32.u10.u12.u20.u22.u23 \tag{e-176}$$
$$-32.u10.u13.u20.u21.u21-32.u10.u13.u20.u22.u22+16.u11.u11.u20.u20.u23-16.u11.u13.u20.u20.u21 \tag{e-177}$$
$$+16.u11.u20.u20.u20.u22-16.u11.u20.u20.u21.u23+16.u12.u12.u20.u20.u23-16.u12.u13.u20.u20.u22 \tag{e-178}$$
$$-16.u12.u20.u20.u20.u21-16.u12.u20.u20.u22.u23+16.u13.u20.u20.u21.u21+16.u13.u20.u20.u22.u22\,]\cdot k \tag{e-179}$$

$$d\cdot\{d,[\{d,s\}^{*},d\,]\}= \tag{e-180}$$
$$+0 \tag{e-181}$$
$$+[+16.u10.u10.u10.u12.u23-16.u10.u10.u10.u13.u22+16.u10.u10.u11.u12.u22+16.u10.u10.u11.u13.u23 \tag{e-182}$$
$$-16.u10.u10.u11.u22.u22-16.u10.u10.u11.u23.u23-16.u10.u10.u12.u12.u21-48.u10.u10.u12.u20.u23 \tag{e-183}$$
$$+16.u10.u10.u12.u21.u22-16.u10.u10.u13.u13.u21+48.u10.u10.u13.u20.u22+16.u10.u10.u13.u21.u23 \tag{e-184}$$
$$-32.u10.u11.u12.u20.u22-32.u10.u11.u13.u20.u23+32.u10.u11.u20.u22.u22+32.u10.u11.u20.u23.u23 \tag{e-185}$$
$$+32.u10.u12.u12.u20.u21+48.u10.u12.u20.u20.u23-32.u10.u12.u20.u21.u22+32.u10.u13.u13.u20.u21 \tag{e-186}$$
$$-48.u10.u13.u20.u20.u22-32.u10.u13.u20.u21.u23+16.u11.u12.u20.u20.u22+16.u11.u13.u20.u20.u23 \tag{e-187}$$
$$-16.u11.u20.u20.u22.u22-16.u11.u20.u20.u23.u23-16.u12.u12.u20.u20.u21-16.u12.u20.u20.u20.u23 \tag{e-188}$$
$$+16.u12.u20.u20.u21.u22-16.u13.u13.u20.u20.u21+16.u13.u20.u20.u20.u22+16.u13.u20.u20.u21.u23\,]\cdot i \tag{e-189}$$
$$+[-16.u10.u10.u10.u11.u23+16.u10.u10.u10.u13.u21-16.u10.u10.u11.u11.u22+16.u10.u10.u11.u12.u21 \tag{e-190}$$
$$+48.u10.u10.u11.u20.u23+16.u10.u10.u11.u21.u22+16.u10.u10.u12.u13.u23-16.u10.u10.u12.u21.u21 \tag{e-191}$$
$$-16.u10.u10.u12.u23.u23-16.u10.u10.u13.u13.u22-48.u10.u10.u13.u20.u21+16.u10.u10.u13.u22.u23 \tag{e-192}$$
$$+32.u10.u11.u11.u20.u22-32.u10.u11.u12.u20.u21-48.u10.u11.u20.u20.u23-32.u10.u11.u20.u21.u22 \tag{e-193}$$
$$-32.u10.u12.u13.u20.u23+32.u10.u12.u20.u21.u21+32.u10.u12.u20.u23.u23+32.u10.u13.u13.u20.u22 \tag{e-194}$$
$$+48.u10.u13.u20.u20.u21-32.u10.u13.u20.u22.u23-16.u11.u11.u20.u20.u22+16.u11.u12.u20.u20.u21 \tag{e-195}$$
$$+16.u11.u20.u20.u20.u23+16.u11.u20.u20.u21.u22+16.u12.u13.u20.u20.u23-16.u12.u20.u20.u21.u21 \tag{e-196}$$
$$-16.u12.u20.u20.u23.u23-16.u13.u13.u20.u20.u22-16.u13.u20.u20.u20.u21+16.u13.u20.u20.u22.u23\,]\cdot j \tag{e-197}$$
$$+[+16.u10.u10.u10.u11.u22-16.u10.u10.u10.u12.u21-16.u10.u10.u11.u11.u23+16.u10.u10.u11.u13.u21 \tag{e-198}$$
$$-48.u10.u10.u11.u20.u22+16.u10.u10.u11.u21.u23-16.u10.u10.u12.u12.u23+16.u10.u10.u12.u13.u22 \tag{e-199}$$
$$+48.u10.u10.u12.u20.u21+16.u10.u10.u12.u22.u23-16.u10.u10.u13.u21.u21-16.u10.u10.u13.u22.u22 \tag{e-200}$$
$$+32.u10.u11.u11.u20.u23-32.u10.u11.u13.u20.u21+48.u10.u11.u20.u20.u22-32.u10.u11.u20.u21.u23 \tag{e-201}$$
$$+32.u10.u12.u12.u20.u23-32.u10.u12.u13.u20.u22-48.u10.u12.u20.u20.u21-32.u10.u12.u20.u22.u23 \tag{e-202}$$
$$+32.u10.u13.u20.u21.u21+32.u10.u13.u20.u22.u22-16.u11.u11.u20.u20.u23+16.u11.u13.u20.u20.u21 \tag{e-203}$$
$$-16.u11.u20.u20.u20.u22+16.u11.u20.u20.u21.u23-16.u12.u12.u20.u20.u23+16.u12.u13.u20.u20.u22 \tag{e-204}$$
$$+16.u12.u20.u20.u20.u21+16.u12.u20.u20.u22.u23-16.u13.u20.u20.u21.u21-16.u13.u20.u20.u22.u22\,]\cdot k \tag{e-205}$$

$$d\cdot\{d,[\{d,s\}^{*},d\,]\}+\{d,[\{d,s\},d\,]\}\cdot d^{*}= \qquad \cdot{*}=\texttt{identity}={*}\cdot \tag{e-206}$$
$$+0 \tag{e-207}$$
$$+0\cdot i \tag{e-208}$$
$$+0\cdot j \tag{e-209}$$
$$+0\cdot k \tag{e-210}$$

$$\boldsymbol{nT3} = [\ (d^*)^2 - d^2] \cdot (\ d^2 \cdot d \cdot [d,s]^* - |d|^2 \cdot [d,s] \cdot d^*\ ) = \tag{e-211}$$
$$+0 \tag{e-212}$$
$$+[+32.u10.u10.u10.u10.u11.u12.u22 + 32.u10.u10.u10.u10.u11.u13.u23 - 32.u10.u10.u10.u10.u11.u22.u22 - \cdots] \cdot i \tag{e-213}$$
$$+[-32.u10.u10.u10.u10.u11.u11.u22 + 32.u10.u10.u10.u10.u11.u12.u21 + 32.u10.u10.u10.u10.u11.u21.u22 + \cdots] \cdot j \tag{e-214}$$
$$+[-32.u10.u10.u10.u10.u11.u11.u23 + 32.u10.u10.u10.u10.u11.u13.u21 + 32.u10.u10.u10.u10.u11.u21.u23 - \cdots] \cdot k \tag{e-215}$$

$$\boldsymbol{nT4} = -[\ d^2 \cdot d \cdot \{d, [\{d,s\}^*, d\,]\} + |d|^2 \cdot \{d, [\{d,s\}, d\,]\} \cdot d^*\ ] = \tag{e-216}$$
$$+0 \tag{e-217}$$
$$+[-32.u10.u10.u10.u10.u11.u12.u22 - 32.u10.u10.u10.u10.u11.u13.u23 + 32.u10.u10.u10.u10.u11.u22.u22 + \cdots] \cdot i \tag{e-218}$$
$$+[+32.u10.u10.u10.u10.u11.u11.u22 - 32.u10.u10.u10.u10.u11.u12.u21 - 32.u10.u10.u10.u10.u11.u21.u22 - \cdots] \cdot j \tag{e-219}$$
$$+[+32.u10.u10.u10.u10.u11.u11.u23 - 32.u10.u10.u10.u10.u11.u13.u21 - 32.u10.u10.u10.u10.u11.u21.u23 + \cdots] \cdot k \tag{e-220}$$

$$\boldsymbol{nT3 + nT4 \equiv 0} \qquad \cdot * = \texttt{identity} = * \cdot \qquad \texttt{numerator}[nar(\lambda = m)] = 0 \qquad \texttt{confirmed!} \tag{e-221}$$

Before we mastered the techniques, and developed the requisite skill, of reasoning with the *native* non-abelian algebra of quaternions, a symbolic calculator was written and used to prove various quaternion identities using component expansions that were too laborious and difficult to do by hand. The provisional unit direction function was generally re-expressed in terms of two distinct parts, $p(\lambda) = ar(\lambda) + nar(\lambda)$, where, $ar(\lambda)$, represented the "*abelian residual*," and, $nar(\lambda)$, was the "*non-abelian residual*." The goal was then to prove that the, $nar(\lambda)$, vanished at the critical point, and/or that the *L'Hôspital's* version, $narh(\lambda)$, did or did not vanish at that point. Various intermediate expression identities needed to be proved, such as:

$$0 \equiv \{d, [\{d,s\}^*, d\,]\} + \{d, [\{d,s\}, d\,]\} \tag{e-222}$$

$$0 \equiv d \cdot \{d, [\{d,s\}^*, d\,]\} + \{d, [\{d,s\}, d\,]\} \cdot d^* \tag{e-223}$$

$$0 \equiv (5d^3[d,s]^* - 3|d|^2[d,s]d^*)|d^2 - (d^*)^2|^2 + (-5d^2 + (d^*)^2)d^3\{d, [\{d,s\}^*, d\,]\} + (-7d^2 + 3(d^*)^2)|d|^2\{d, [\{d,s\}, d\,]\}d^* \tag{e-224}$$

$$0 \equiv ((d^*)^2 - d^2)(\ d^3[d,s]^* - |d|^2[d,s]d^*\ ) - [\ d^3\{d, [\{d,s\}^*, d\,]\} + |d|^2\{d, [\{d,s\}, d\,]\}d^*\ ] \tag{e-225}$$

Once we mastered the native algebra methods, the component expansion procedures became somewhat obsolete, and the text of this paper now includes the native reasoning used to prove these things, in a more efficient manner, often taking completely different derivation paths than we took before when employing this calculator. Hence, some of these identities may not be found in the current text. However, the code can still be used to check or confirm quaternion expression identities that are difficult to prove. It is especially useful to confirm first, with the symbolic calculator, that the identity is true, before attempting to discover the native algebra proof for it.

The above quaternion component expansions are sample outputs from the symbolic calculator used to verify the vanishing of terms in order to justify applying *L'Hospital's Rule* to the provisional unit direction function, $p(\lambda)$. Not all outputs are shown, nor are all those shown printed with all terms given. However, the "*C Source code*" for the *Hypercomplex Symbolic Expression Calculator for Quaternions*, used here to produce this output, is given below. The file name is `hs.c`. The reader may compile the code and run it to see all the remaining outputs.

Since this part of the work is considered "depreciated" content, no further explanations are given on the use of this code. It it presented "as is," in case some readers, who are well acquainted with c-code, and thus can read and understand what it does on their own, may find it useful. For the remaining readers, we refer to the native derivations in the body of the text given above, as the current method of reckoning being promoted by this paper.

C Source code:

```
/* hs.c : hypercomplex symbolic expression calculator for quaternions
          "a minimal calculator": add, sub, mul, smul, conj.
p(r=m) = (u1 + u2)/2 + nar(r=m)
P(r=m) = (u1 + u2)/2 + narh(r=m)
Evaluating: nar(r), narh(r)  -- "Non-Abelian Residual Function" and "L'Hospital's version"
Ref eqn: aq^2 + bq + c = 0

narh(r=m) =  [ nT1 + nT2 ]/[ 8.|d|^4.|(d^2 - (d*)^2)|^2 ]
nT1 = [ 5.d^2.d.[d,s]* - 3.|d|^2.[d,s].d* ].|(d^2 - (d*)^2)|^2
nT2 = [ (-5.d^2 + (d*)^2).d^2.d.{d,[{d,s}*,d]} + (-7.d^2 + 3.(d*)^2).|d|^2.{d,[{d,s},d]}.d* ]
Proving that the "whole term" vanishes... nT1 + nT2 = 0

nar(r=m) = ...( nT3 + nT4 )/[ |d|^4.(r^4 - m^4) ]...with, (r=m)
nT3 = [ (d*)^2 - d^2 ].(d^2.d.[d,s]* - |d|^2.[d,s].d*)
nT4 = -(d^2.d.{d,[{d,s}*,d]} + |d|^2.{d,[{d,s},d]}.d*)
Proving that the "numerator" vanishes... nT3 + nT4 = 0
*/
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <string.h>

typedef struct term { int n; char *ps; } term_s;
typedef struct expr { term_s tm; struct expr *prv; struct expr *nxt; } expr_s;
typedef struct quat { expr_s *t, *x, *y, *z; } quat_s;

int updateTerm(term_s *tm, int n, char *ps );
int printTerm(term_s *tm);
int sortps(char *ps);  // sorts product string, e.g. "u20.u11.u10.u21" --> "u10.u11.u20.u21"
void error(char *fn, char *msg);   // minimal error handling: on any error we exit pgm with msg.

expr_s * newExprTerm(expr_s *prv, expr_s *nxt);
int delExprTerm(expr_s **ex);
int printExpr(expr_s *ex);
int addExpr(expr_s *de, expr_s *e1, expr_s *e2);
int subExpr(expr_s *de, expr_s *e1, expr_s *e2);
int smulExpr(expr_s *ex, int s);  // multiply throughout by a scalar s
int mulExpr(expr_s *de, expr_s *e1, expr_s *e2);
int reduceExpr(expr_s **ex);
int purgeExpr(expr_s **ex);
int sortExpr(expr_s *ex); // sorts expr list, e.g. "u20.u21 + u10.u11" --> "u10.u11 + u20.u21"
int cleanExpr(int n, ...); // clean "n" variable args of type:  expr_s *

int initQuat(term_s *t, term_s *x, term_s *y, term_s *z, int n, ...); // "n" var args of type: quat_s *
int printQuat(quat_s *q);
int copyQuat(quat_s *dq, quat_s *sq);
int conjQuat(quat_s *q);
int addQuat(quat_s *qd, quat_s *q1, quat_s *q2);
int subQuat(quat_s *qd, quat_s *q1, quat_s *q2);
int smulQuat(quat_s *q, int s);  // multiply quat by a scalar s
int mulQuat(quat_s *qd, quat_s *q1, quat_s *q2);
int reduceQuat(quat_s *q);
int purgeQuat(quat_s *q);
int sortQuat(quat_s *q);

/* Linked-List Expr Model
   ======================
   An expression is a linked-list of expr nodes, where each node contains a term "tm,"
   and pointers to previous and next nodes. Each term consists of an integer number "n,"
   and a product string "ps," which is a character string like "u10.u11.u20". These
   product strings are sorted by "sortps()" to all have one unique representation.
   So "u20.u11.u10" and "u11.u20.u10" and "u10.u11.u20" are all represented by one
   unique form "u10.u11.u20"..etc..The sign of the integer "n" determines whether the
   term is add "+" (n >=0) , or sub "-" (n <0), and the special symbol "#" is used
   to represent the null term, "+0.#", that has zero numeric multiplier.

   'addExpr': simply appends one list to another and returns the result in the destination expr,
    which can be the same as one of the source exprs.
   'reduceExpr': then combines all the terms found within an expression that have same product string,
    hence some will result in zero numeric multiplier (n=0) when opposite signs combine.
   'purgeExpr': then removes all zeros (n=0) from the expr list, except the first expr node,
    which, if 0, is then set to read "+0.#" to keep the first node always ready for re-use.
   'sortExpr': then re-arranges the expr into an unique order of summation, so that two differently
    obtained exprs can be easily compared by eye for equality.
   'cleanExpr': removes all the child nodes of a head expr, freeing up the memory, while
    keeping the first node inviolate, and ready for re-use.

    quat_s is then a structure of 4 symbolic expressions t,x,y,z. The expressions are all abelian,
    being composed of the real variable components of quaternions. The non-abelian characteristic
    enters into the calculations through the sole function mulQuat(), which alone encapsulates all
    the non-commuting property of the quaternions.
*/
```

```
int main( void )
{
   term_s tm_t, tm_x, tm_y, tm_z;
   quat_s u1, u2, w1, w2, w3, w4;

   // declare the quaternion variables */
   //     name                    description
   quat_s s;                /*  = u1 + u2                          */
   quat_s d;                /*  = u1 - u2                          */
   quat_s sc;               /*  = s* = u1* + u2*                   */
   quat_s dc;               /*  = d* = u1* - u2*                   */
   quat_s ds;               /*  = (u1 - u2).(u1 + u2)              */
   quat_s sd;               /*  = (u1 + u2).(u1 - u2)              */
   quat_s dspsd;            /*  = {d,s}                            */
   quat_s dsmsd;            /*  = [d,s]                            */
   quat_s dspsdc;           /*  = {d,s}*                           */
   quat_s dsmsdc;           /*  = [d,s]*                           */
   quat_s dd;               /*  = d^2                              */
   quat_s dcdc;             /*  = (d*)^2                           */
   quat_s ddc;              /*  = |d|^2                            */
   quat_s dsmsd_dc;         /*  = [d,s].d*                         */
   quat_s d_dsmsdc;         /*  = d.[d,s]*                         */
   quat_s dd_d_dsmsdc;      /*  = d^2.d.[d,s]*                     */
   quat_s ddc_dsmsd_dc;     /*  = |d|^2.[d,s].d*                   */
   quat_s dd_d_dsmsdc_5;    /*  = 5.d^2.d.[d,s]*                   */
   quat_s ddc_dsmsd_dc_3;   /*  = 3.|d|^2.[d,s].d*                 */
   quat_s v1;               /*  = 5.d^2.d.[d,s]* - 3.|d|^2.[d,s].d*   */
   quat_s ddmdcdc;          /*  = d^2 - (d*)^2                     */
   quat_s dcdcmdd;          /*  = (d*)^2 - d^2                     */
   quat_s v2;               /*  = (d^2 - (d*)^2).((d*)^2 - d^2)    */


   quat_s nT1;              /*  = v1.v2
                                = [ 5.d^2.d.[d,s]* - 3.|d|^2.[d,s].d* ]
                                 .[ (d^2 - (d*)^2).((d*)^2 - d^2) ]    */


   quat_s dspsd_d;          /*  = [{d,s},d]              */
   quat_s dspsdc_d;         /*  = [{d,s}*,d]             */
   quat_s ad_dspsd_d;       /*  = {d,[{d,s},d]}          */
   quat_s ad_dspsdc_d;      /*  = {d,[{d,s}*,d]}         */
   quat_s ad_dspsd_d_dc;    /*  = {d,[{d,s},d]}.d*       */
   quat_s d_ad_dspsdc_d;    /*  = d.{d,[{d,s}*,d]}       */
   quat_s dd_d_ad_dspsdc_d; /*  = d^2.d.{d,[{d,s}*,d]}   */
   quat_s ddc_ad_dspsd_d_dc;/*  = |d|^2.{d,[{d,s},d]}.d*  */
   quat_s dd_5;             /*  = 5.d^2                  */
   quat_s dd_7;             /*  = 7.d^2                  */
   quat_s dcdc_3;           /*  = 3.(d*)^2               */
   quat_s dcdc_dd_5;        /*  = -5.d^2 + (d*)^2        */
   quat_s dcdc_3_dd_7;      /*  = -7.d^2 + 3.(d*)^2      */
   quat_s v3;               /*  = (-5.d^2 + (d*)^2).d^3.{d,[{d,s}*,d]}         */
   quat_s v4;               /*  = (-7.d^2 + 3.(d*)^2).|d|^2.{d,[{d,s},d]}.d*   */


   quat_s nT2;              /*  =  v3 + v4
                                = (-5.d^2 + (d*)^2).d^3.{d,[{d,s}*,d]}
                                +   (-7.d^2 + 3.(d*)^2).|d|^2.{d,[{d,s},d]}.d*  */

   quat_s nT1_plus_nT2;     /*  = nT1 + nT2              */

   quat_s nT3;              /*  = [ (d*)^2 - d^2 ].(d^2.d.[d,s]* - |d|^2.[d,s].d*)        */
   quat_s nT4;              /*  = nT4 = -(d^2.d.{d,[{d,s}*,d]} + |d|^2.{d,[{d,s},d]}.d*)   */

   quat_s nT3_plus_nT4;     /*  = nT3 + nT4              */

   tm_t.ps = tm_x.ps = tm_y.ps = tm_z.ps = NULL;

   // make: u1
   updateTerm(&tm_t,1,"u10");
   updateTerm(&tm_x,1,"u11");
   updateTerm(&tm_y,1,"u12");
   updateTerm(&tm_z,1,"u13");

   initQuat(&tm_t,&tm_x,&tm_y,&tm_z,1,&u1);

   // make: u2
   updateTerm(&tm_t,1,"u20");
   updateTerm(&tm_x,1,"u21");
   updateTerm(&tm_y,1,"u22");
   updateTerm(&tm_z,1,"u23");

   initQuat(&tm_t,&tm_x,&tm_y,&tm_z,1,&u2);
```

```
// make: default "+0.#" term
updateTerm(&tm_t,0,"#");
updateTerm(&tm_x,0,"#");
updateTerm(&tm_y,0,"#");
updateTerm(&tm_z,0,"#");

// initialize all quaternion variables before first use
// make: all 48 remaining
initQuat(&tm_t,&tm_x,&tm_y,&tm_z,48,
         &s,
         &d,
         &sc,
         &dc,
         &ds,
         &sd,
         &dspsd,
         &dsmsd,
         &dspsdc,
         &dsmsdc,

         &dd,
         &dcdc,
         &ddc,
         &dsmsd_dc,
         &d_dsmsdc,
         &dd_d_dsmsdc,
         &ddc_dsmsd_dc,
         &dd_d_dsmsdc_5,
         &ddc_dsmsd_dc_3,
         &v1,

         &ddmdcdc,
         &dcdcmdd,
         &v2,
         &nT1,
         &dspsd_d,
         &dspsdc_d,
         &ad_dspsd_d,
         &ad_dspsdc_d,
         &ad_dspsd_d_dc,
         &d_ad_dspsdc_d,

         &dd_d_ad_dspsdc_d,
         &ddc_ad_dspsd_d_dc,
         &dd_5,
         &dd_7,
         &dcdc_3,
         &dcdc_dd_5,
         &dcdc_3_dd_7,
         &v3,
         &v4,
         &nT2,

         &nT1_plus_nT2,
         &w1,
         &w2,
         &w3,
         &w4,
         &nT3,
         &nT4,
         &nT3_plus_nT4

);


// calculate the symbolic expressions for each quaternion variable
// nT1 .. calculations:
addQuat(&s,&u1,&u2);
subQuat(&d,&u1,&u2);
copyQuat(&sc,&s); conjQuat(&sc);
copyQuat(&dc,&d); conjQuat(&dc);
mulQuat(&ds,&d,&s);
mulQuat(&sd,&s,&d);
addQuat(&dspsd,&ds,&sd);
subQuat(&dsmsd,&ds,&sd);
copyQuat(&dspsdc,&dspsd); conjQuat(&dspsdc);
copyQuat(&dsmsdc,&dsmsd); conjQuat(&dsmsdc);
mulQuat(&dd,&d,&d);
mulQuat(&dcdc,&dc,&dc);
mulQuat(&ddc,&d,&dc);
mulQuat(&dsmsd_dc,&dsmsd,&dc);
mulQuat(&d_dsmsdc,&d,&dsmsdc);
```

```
mulQuat(&dd_d_dsmsdc,&dd,&d_dsmsdc);
mulQuat(&ddc_dsmsd_dc,&ddc,&dsmsd_dc);
copyQuat(&dd_d_dsmsdc_5,&dd_d_dsmsdc);   smulQuat(&dd_d_dsmsdc_5,5);
copyQuat(&ddc_dsmsd_dc_3,&ddc_dsmsd_dc); smulQuat(&ddc_dsmsd_dc_3,3);
subQuat(&v1,&dd_d_dsmsdc_5,&ddc_dsmsd_dc_3);
subQuat(&ddmdcdc,&dd,&dcdc);
subQuat(&dcdcmdd,&dcdc,&dd);
mulQuat(&v2,&ddmdcdc,&dcdcmdd);
// nT1 = v1.v2 = [ 5.d^2.d.[d,s]* - 3.|d|^2.[d,s].d* ].[ (d^2 - (d*)^2).((d*)^2 - d^2) ]  */
mulQuat(&nT1,&v1,&v2);

// nT2 .. calculations:
mulQuat(&w1,&dspsd,&d);    mulQuat(&w2,&d,&dspsd);    subQuat(&dspsd_d,&w1,&w2);
mulQuat(&w1,&dspsdc,&d);   mulQuat(&w2,&d,&dspsdc);   subQuat(&dspsdc_d,&w1,&w2);
mulQuat(&w1,&d,&dspsd_d);  mulQuat(&w2,&dspsd_d,&d);  addQuat(&ad_dspsd_d,&w1,&w2);
mulQuat(&w1,&d,&dspsdc_d); mulQuat(&w2,&dspsdc_d,&d); addQuat(&ad_dspsdc_d,&w1,&w2);
mulQuat(&ad_dspsd_d_dc,&ad_dspsd_d,&dc);
mulQuat(&d_ad_dspsdc_d,&d,&ad_dspsdc_d);
mulQuat(&dd_d_ad_dspsdc_d,&dd,&d_ad_dspsdc_d);
mulQuat(&ddc_ad_dspsd_d_dc,&ddc,&ad_dspsd_d_dc);
copyQuat(&dd_5,&dd); smulQuat(&dd_5,5);
copyQuat(&dd_7,&dd); smulQuat(&dd_7,7);
copyQuat(&dcdc_3,&dcdc); smulQuat(&dcdc_3,3);
subQuat(&dcdc_dd_5,&dcdc,&dd_5);
subQuat(&dcdc_3_dd_7,&dcdc_3,&dd_7);
mulQuat(&v3,&dcdc_dd_5,&dd_d_ad_dspsdc_d);
mulQuat(&v4,&dcdc_3_dd_7,&ddc_ad_dspsd_d_dc);
// nT2  =  v3 + v4 = (-5.d^2 + (d*)^2).d^3.{d,[{d,s}*,d]} + (-7.d^2 + 3.(d*)^2).|d|^2.{d,[{d,s},d]}.d*  */
addQuat(&nT2,&v3,&v4);


 // narh(r=m):
 // checking that nT1 + nT2 = 0
 addQuat(&nT1_plus_nT2,&nT1,&nT2);   /* narh(r=m) = 0  check point! */


// nT3 .. calculations:
subQuat(&w4,&dd_d_dsmsdc,&ddc_dsmsd_dc);
// nT3 =   + [ (d*)^2 - d^2 ].( d^2.d.[d,s]* - |d|^2.[d,s].d* )
mulQuat(&nT3,&dcdcmdd,&w4);


// nT4 .. calculations:
addQuat(&nT4,&dd_d_ad_dspsdc_d,&ddc_ad_dspsd_d_dc);
// nT4 =   - ( d^3.{d,[{d,s}*,d]} + |d|^2.{d,[{d,s},d]}.d* )
smulQuat(&nT4,-1);


 // nar(r=m):
 // checking that nT3 + nT4 = 0
 addQuat(&nT3_plus_nT4,&nT3,&nT4);   /* Numerator[nar(r=m)] = 0  check point! */



// print out the symbolic expressions for each quaternion variable

printf("u1 = \n");                          printQuat(&u1);
printf("u2 = \n");                          printQuat(&u2);
printf("s = \n");                           printQuat(&s);
printf("d = \n");                           printQuat(&d);
printf("sc = \n");                          printQuat(&sc);
printf("dc = \n");                          printQuat(&dc);
printf("ds = \n");                          printQuat(&ds);
printf("sd = \n");                          printQuat(&sd);
printf("dspsd = \n");                       printQuat(&dspsd);
printf("dsmsd = \n");                       printQuat(&dsmsd);
printf("dspsdc = \n");                      printQuat(&dspsdc);
printf("dsmsdc = \n");                      printQuat(&dsmsdc);
printf("dd = \n");                          printQuat(&dd);
printf("dcdc = \n");                        printQuat(&dcdc);
printf("ddc = \n");                         printQuat(&ddc);
printf("dsmsd_dc = \n");                    printQuat(&dsmsd_dc);
printf("d_dsmsdc = \n");                    printQuat(&d_dsmsdc);
printf("dd_d_dsmsdc = \n");                 printQuat(&dd_d_dsmsdc);
printf("ddc_dsmsd_dc = \n");                printQuat(&ddc_dsmsd_dc);
printf("dd_d_dsmsdc_5 = \n");               printQuat(&dd_d_dsmsdc_5);
printf("ddc_dsmsd_dc_3 = \n");              printQuat(&ddc_dsmsd_dc_3);
printf("v1 = \n");                          printQuat(&v1);
printf("ddmdcdc = \n");                     printQuat(&ddmdcdc);
printf("dcdcmdd = \n");                     printQuat(&dcdcmdd);
printf("v2 = \n");                          printQuat(&v2);
printf("nT1 = \n");                         printQuat(&nT1);
printf("dspsd = \n");                       printQuat(&dspsd);
```

```
   printf("dspsdc = \n");                    printQuat(&dspsdc);
   printf("dspsd_d = \n");                   printQuat(&dspsd_d);
   printf("dspsdc_d = \n");                  printQuat(&dspsdc_d);
   printf("ad_dspsd_d = \n");                printQuat(&ad_dspsd_d);
   printf("ad_dspsdc_d = \n");               printQuat(&ad_dspsdc_d);
   printf("ad_dspsd_d_dc = \n");             printQuat(&ad_dspsd_d_dc);
   printf("d_ad_dspsdc_d = \n");             printQuat(&d_ad_dspsdc_d);
   printf("dd_d_ad_dspsdc_d = \n");          printQuat(&dd_d_ad_dspsdc_d);
   printf("ddc_ad_dspsd_d_dc = \n");         printQuat(&ddc_ad_dspsd_d_dc);
   printf("dd_5 = \n");                      printQuat(&dd_5);
   printf("dd_7 = \n");                      printQuat(&dd_7);
   printf("dcdc_3 = \n");                    printQuat(&dcdc_3);
   printf("dcdc_dd_5 = \n");                 printQuat(&dcdc_dd_5);
   printf("dcdc_3_dd_7 = \n");               printQuat(&dcdc_3_dd_7);
   printf("v3 = \n");                        printQuat(&v3);
   printf("v4 = \n");                        printQuat(&v4);
   printf("nT2 = \n");                       printQuat(&nT2);
   printf("nT1_plus_nT2 = \n");              printQuat(&nT1_plus_nT2);
   printf("nT3 = \n");                       printQuat(&nT3);
   printf("nT4 = \n");                       printQuat(&nT4);
   printf("nT3_plus_nT4 = nT3 + nT4 = \n");  printQuat(&nT3_plus_nT4);


   // calc and print the two identity relations:
   addQuat(&w1,&ad_dspsdc_d,&ad_dspsd_d);
   printf("ad_dspsdc_d + ad_dspsd_d = {d,[{d,s}*,d]} + {d,[{d,s},d]} = ... \n");
   printQuat(&w1);

   addQuat(&w2,&d_ad_dspsdc_d,&ad_dspsd_d_dc);
   printf("d_ad_dspsdc_d + ad_dspsd_d_dc = d.{d,[{d,s}*,d]} + {d,[{d,s},d]}.d*  =  \n");
   printQuat(&w2);

//   addQuat(&w3,&dd_d_ad_dspsdc_d,&ddc_ad_dspsd_d_dc);
//   printf("ddc_ad_dspsd_d_dc + dd_d_ad_dspsdc_d = |d|^2.{d,[{d,s},d]}.d* + d^2.d.{d,[{d,s}*,d]} =  \n");
//   printQuat(&w3);


   printf("\n\n all done!\n\n");

    return 0;
}


void error(char *fn, char *msg)
{
    printf("%s : %s\n",fn,msg);
    exit(0);
}

int updateTerm(term_s *tm, int n, char *ps)
{
    if ( tm == NULL )
      error("updateTerm","tm is null");

    if ( ps == NULL )
      {
         if ( tm->ps != NULL )
           {
              free(tm->ps);
              tm->ps = NULL;
           }
      }
    else  /* ps != NULL */
      {
         if ( (tm->ps != NULL) && (strlen(tm->ps) != strlen(ps)) )
            {
               free(tm->ps);
               tm->ps = NULL;
            }
         if ( tm->ps == NULL )
               if ( (tm->ps = (char *) calloc(strlen(ps)+1,sizeof(char))) == NULL )
                     error("updateTerm","can't alloc new ps");
         strcpy(tm->ps,ps);
      }

    tm->n = n;

    return 0;

}

int printTerm(term_s *tm)
{
```

```
    char sign = '+';
    int n;

    if ( tm == NULL )
     {
       printf("(NULL) ");
       return 0;
     }

    if ( (n = tm->n) < 0 ) { sign = '-'; n *= -1; }

    if ( tm->ps == NULL )
        printf("%c %d.%s ",sign,n,"(null)");
    else if (strlen(tm->ps) == 0 )
        printf("%c %d.%s ",sign,n,"(empty)");
    else
        printf("%c %d.%s ",sign,n,tm->ps);

    return 0;
}

int sortps(char *ps)
{
   char *tps = NULL;
   char *nps = NULL;
   char *mintoken = NULL;
   char *p = NULL;
   char *p_start = NULL;
   char *p_end = NULL;
   char *p_mintoken = NULL;
   char  savechr;
   int  ndots = 0;
   int  ntokens = 0;

   if ( ps == NULL )
      return 0;
   if (strlen(ps) == 0)
      return 0;

    /* e.g. "u20.u21.u11.u10" --> "u10.u11.u20.u21" */



    if ( (tps = (char *) calloc(strlen(ps) + 1, sizeof(char))) == NULL )
       error("sortps","can't alloc mem for tps");
    if ( (nps = (char *) calloc(strlen(ps) + 1, sizeof(char))) == NULL )
       error("sortps","can't alloc mem for nps");

    strcpy(tps,ps);
    nps[0] = '\0';

    p = tps;
    while ( *p != '\0' )
       if ( *p++ == '.' ) ndots++;


    while( strlen(tps) > 0 )
     {

         if ( ++ntokens > (ndots + 1) )
            error("sortps","ndots token count wrong");

        /* find the smallest token in tps,
           copy it,
           append it into nps,
           delete it from tps.
        */

         p = tps;
         p_end = p;

         while ( *p != '\0' )
         {
         /* find next token */
         p_start = p;
         while ( *p != '\0' && *p != '.') p++;
         p_end = p;

         savechr = *p_end;
         *p_end = '\0';

         if ( strlen(p_start) == 0 )
              error("sortps","p_start is empty str");
```

```
          if ( (mintoken == NULL) || (strcmp(mintoken,p_start) > 0) )
             {
               if ( mintoken != NULL )
                 {
                    free(mintoken);
                    mintoken = NULL;
                 }
               if ( (mintoken = (char *) calloc(strlen(p_start) + 1, sizeof(char))) == NULL )
                  error("sortps","can't alloc token mem");
               strcpy(mintoken,p_start);
               p_mintoken = p_start;
             }
           *p_end = savechr;
           if ( *p == '.' ) p++;
         }  /* end while ( *p != '\0' ) */

         /* we now have the min token in tps */
         /* so append it to nps */
         if ( mintoken == NULL )
           error("sortps","mintoken null in wrong place");

         if ( strlen(nps) == 0 )
            strcat(nps,mintoken);
         else
            {
            strcat(nps,"."); strcat(nps,mintoken);
            }

         /* now delete the mintoken from tps */
         p = p_mintoken;
         p_start = p;
         while ( *p != '\0' && *p != '.' ) p++;
         p_end = p;
         if ( *p_end == '\0' )
           {
             *p_start = '\0'; /* remove last token */
              if ( tps[strlen(tps)] == '.' )
                  tps[strlen(tps)] = '\0';  /* remove last dot if present */
           }
          else /* *p_end == '.' */
           {
              p++; /* advance to start of next token */
              while ( (*p_start = *p) != '\0' ) { p_start++; p++; } /* copy rem str incl last '\0' */
           }
          if (mintoken != NULL )
           {
              free(mintoken);
              mintoken = NULL;
           }

         /* so we've removed mintoken from tps, repeat to get next mintoken from remaining tps */
     }

     strcpy(ps,nps); /* copy over the sorted string to input ps mem location */
     if ( tps != NULL )   {  free(tps); tps = NULL; };
     if ( nps != NULL )   {  free(nps); nps = NULL; };
     if ( mintoken != NULL ) { free(mintoken); mintoken = NULL; };

     return 0;
}


expr_s * newExprTerm(expr_s *prv, expr_s *nxt)
{
     expr_s *theNew;

     if ( (theNew = (expr_s *) calloc(1,sizeof(expr_s))) == NULL )
         error("newExprTerm","calloc memory failed");
     theNew->prv = prv;
     theNew->nxt = nxt;
     theNew->tm.ps = NULL;
     return theNew;
}


int printExpr(expr_s *ex)
{
   expr_s *curExprTerm;
   expr_s *prv;
   expr_s *nxt;

   curExprTerm = ex;
```

```
   if (curExprTerm == NULL )
    {
       printf("(Null) ");
       return 0;
    }

   /* find the start of the expression list */
   while (curExprTerm->prv != NULL )
    {
     prv = curExprTerm->prv;
     curExprTerm = prv;
     }
   /* now print all terms from beginning to end of list */
   while (curExprTerm != NULL )
    {
      printTerm(&(curExprTerm->tm));
      nxt = curExprTerm->nxt;
      curExprTerm = nxt;
     }
   printf("\n");
   return 0;

}


int addExpr(expr_s *de, expr_s *e1, expr_s *e2)
{
     expr_s *thisNew = NULL;
     expr_s *thisOld = NULL;
     expr_s *thisCur = NULL;
     expr_s *prv = NULL;
     expr_s *nxt = NULL;
     expr_s *prvNew = NULL;
     expr_s *nxtNew = NULL;


     thisOld = e1;
     if ( thisOld != NULL )
      {
         /* go to the start of the old expression e1 */
         while (thisOld->prv != NULL )
          {
            prv = thisOld->prv;
            thisOld = prv;
          }
        /* now add all terms from e1 to new list */

         while (thisOld != NULL )
          {
             if ( (thisNew = newExprTerm(prvNew, (expr_s *)NULL)) == NULL )
               error("addExpr","failed to get NewExprTerm 1");

             updateTerm(&(thisNew->tm), thisOld->tm.n, thisOld->tm.ps);
             if ( thisNew->prv != NULL )
                  thisNew->prv->nxt = thisNew;
             prvNew = thisNew;
             nxt = thisOld->nxt;
             thisOld = nxt;

          }
       }

     thisOld = e2;
     if ( thisOld != NULL )
      {
         /* go to the start of the old expression e2 */
         while (thisOld->prv != NULL )
          {
            prv = thisOld->prv;
            thisOld = prv;
          }
        /* now add all terms from e2 to new list */

         while (thisOld != NULL )
          {
             if ( (thisNew = newExprTerm(prvNew, (expr_s *)NULL)) == NULL )
               error("addExpr","failed to get NewExprTerm 2");

             updateTerm(&(thisNew->tm), thisOld->tm.n, thisOld->tm.ps);
             if ( thisNew->prv != NULL )
                  thisNew->prv->nxt = thisNew;
             prvNew = thisNew;
             nxt = thisOld->nxt;
```

```
            thisOld = nxt;

          }
       }

     if ( thisNew == NULL )
        error("addExpr","nothing added in this sum!");

     /* thisNew is currently pointing to end of expr sum list so backup to start */
     while (thisNew->prv != NULL)
       {
         prv = thisNew->prv;
         thisNew = prv;
        }

    if ( de == NULL )
       error("addExpr","de is null");



      /* we "re-use" the first node of destination expr, but remove all it's remaining original entries */
      /* it is an error to call addExpr with non-null de pointing to any other but 1st node  */

      if ( de->prv != NULL )
         error("addExpr","de must be null or point to 1st node in expression!");

       /* now clean all terms but 1st out of this de expr */
      thisCur = de->nxt;

      while (thisCur != NULL)
       {
         nxt = thisCur->nxt;
         delExprTerm(&thisCur);
         thisCur = nxt;
       }

       de->nxt = thisNew;
       thisNew->prv = de;
       updateTerm( &(de->tm), 0,"#");
#if 0
       printf("HERER...\n");
       printExpr(de);
       printf("DONE...\n");
#endif

#if 0
       /* then let de expr start replace thisNew start */
       de->nxt = thisNew->nxt;
       updateTerm(&(de->tm),thisNew->tm.n,thisNew->tm.ps);
       if ( thisNew->nxt != NULL ) thisNew->nxt->prv = de;
       thisNew->prv = NULL;
       thisNew->nxt = NULL;
       if ( thisNew->tm.ps != NULL )
         {
               free(thisNew->tm.ps); thisNew->tm.ps = NULL;
         }
       free(thisNew);
       thisNew = NULL;

#endif

    return 0;
}


int delExprTerm(expr_s **ex)
{
   expr_s *prv;
   expr_s *nxt;

   if ( ex == NULL || *ex == NULL )
     return 0;
   prv = (*ex)->prv;
   nxt = (*ex)->nxt;
   if ( (*ex)->tm.ps != NULL )
     {
       free( (*ex)->tm.ps );
       (*ex)->tm.ps = NULL;
      }
   free( *ex );
   *ex = NULL;
   if ( prv != NULL )
     prv->nxt = nxt;
```

```
   if ( nxt != NULL )
     nxt->prv = prv;
   return 0;
}


int subExpr(expr_s *de, expr_s *e1, expr_s *e2)
{

     expr_s *thisNew = NULL;
     expr_s *thisOld = NULL;
     expr_s *thisCur = NULL;
     expr_s *prv;
     expr_s *nxt;
     expr_s *prvNew = NULL;
     expr_s *nxtNew = NULL;


     thisOld = e1;
     if ( thisOld != NULL )
      {
         /* go to the start of the old expression e1 */
         while (thisOld->prv != NULL )
          {
            prv = thisOld->prv;
            thisOld = prv;
          }
        /* now add all terms from e1 to new list */

         while (thisOld != NULL )
          {
             if ( (thisNew = newExprTerm(prvNew, (expr_s *)NULL)) == NULL )
               error("addExpr","failed to get NewExprTerm 1");

             updateTerm(&(thisNew->tm), thisOld->tm.n, thisOld->tm.ps);
             if ( thisNew->prv != NULL )
                  thisNew->prv->nxt = thisNew;
             prvNew = thisNew;
             nxt = thisOld->nxt;
             thisOld = nxt;

          }
       }

     thisOld = e2;
     if ( thisOld != NULL )
      {
         /* go to the start of the old expression e2 */
         while (thisOld->prv != NULL )
          {
            prv = thisOld->prv;
            thisOld = prv;
          }
        /* now add all terms from e2 to new list */

         while (thisOld != NULL )
          {
             if ( (thisNew = newExprTerm(prvNew, (expr_s *)NULL)) == NULL )
               error("addExpr","failed to get NewExprTerm 2");

             updateTerm(&(thisNew->tm), thisOld->tm.n, thisOld->tm.ps);
             /* change sign on this expression terms */
              thisNew->tm.n *= -1;

             if ( thisNew->prv != NULL )
                  thisNew->prv->nxt = thisNew;
             prvNew = thisNew;
             nxt = thisOld->nxt;
             thisOld = nxt;

          }
       }

     if ( thisNew == NULL )
        error("addExpr","nothing added in this sum!");

     /* thisNew is currently pointing to end of expr sum list so backup to start */
     while (thisNew->prv != NULL)
       {
         prv = thisNew->prv;
         thisNew = prv;
        }
```

```
     if ( de == NULL )
         error("subExpr","de is NULL");


         /* we "re-use" the first node of destination expr, but remove all it's remaining original entries */
         /* it is an error to call addExpr with non-null de pointing to any other but 1st node  */
         if ( de->prv != NULL )
            error("addExpr","de must be null or point to 1st node in expression!");

         /* now clean all terms but 1st out of this de expr */

         thisCur = de->nxt;
         while (thisCur != NULL)
          {
            nxt = thisCur->nxt;
            delExprTerm(&thisCur);
            thisCur = nxt;
          }

       /* then let de expr start replace thisNew start */
       de->nxt = thisNew->nxt;
       updateTerm(&(de->tm),thisNew->tm.n,thisNew->tm.ps);
       if ( thisNew->nxt != NULL ) thisNew->nxt->prv = de;
       thisNew->prv = NULL;
       thisNew->nxt = NULL;
       if ( thisNew->tm.ps != NULL )
         {
               free(thisNew->tm.ps); thisNew->tm.ps = NULL;
         }
       free(thisNew);
       thisNew = NULL;


    return 0;

}


int smulExpr(expr_s *ex, int s)
{
    expr_s *cur;

    for ( cur = ex; cur != NULL; cur = cur->nxt )
       cur->tm.n *= s;

    return 0;
}


int mulExpr(expr_s *de, expr_s *e1, expr_s *e2)
{
     expr_s *thisNew = NULL;
     expr_s *thisOld1 = NULL;
     expr_s *thisOld2 = NULL;
     expr_s *prv1 = NULL;
     expr_s *nxt1 = NULL;
     expr_s *prv2 = NULL;
     expr_s *nxt2 = NULL;


     expr_s *thisCur = NULL;
     expr_s *prv;
     expr_s *nxt;
     expr_s *prvNew = NULL;
     expr_s *nxtNew = NULL;
     char * tmpstr = NULL;

     thisOld1 = e1;
     thisOld2 = e2;
     while (thisOld1 != NULL)
     {
          while (thisOld2 != NULL)
            {
             if ( (thisNew = newExprTerm(prvNew, (expr_s *)NULL)) == NULL )
               error("nulExpr","failed to get NewExprTerm 1");

              if ( thisOld1->tm.ps == NULL )
               error("mulExpr","nul string in thisOld1");
              if ( thisOld2->tm.ps == NULL )
               error("mulExpr","nul string in thisOld2");
```

```
            tmpstr = (char *) calloc(strlen(thisOld1->tm.ps)+strlen(thisOld2->tm.ps)+2,sizeof(char));
            if ( tmpstr == NULL )
               error("mulExpr","out of memory");

            tmpstr[0]='\0';
            strcat(tmpstr,thisOld1->tm.ps);
            strcat(tmpstr,".");
            strcat(tmpstr,thisOld2->tm.ps);
            updateTerm(&(thisNew->tm), (thisOld1->tm.n) * (thisOld2->tm.n),tmpstr);
            free(tmpstr);
            tmpstr = NULL;

            if ( thisNew->prv != NULL )
                 thisNew->prv->nxt = thisNew;
            prvNew = thisNew;
            nxt2 = thisOld2->nxt;
            thisOld2 = nxt2;
           }
         thisOld2 = e2;
         nxt1 = thisOld1->nxt;
         thisOld1 = nxt1;

     }

     if ( thisNew == NULL )
        error("addExpr","nothing added in this mul!");

     /* thisNew is currently pointing to end of expr sum list so backup to start */
     while (thisNew->prv != NULL)
       {
         prv = thisNew->prv;
         thisNew = prv;
        }


     if ( de == NULL )
      {
         de = thisNew;
      }
      else /* ( de != NULL ) */
       {
         /* we "re-use" the first node of destination expr, but remove all it's remaining original entries */
         /* it is an error to call addExpr with non-null de pointing to any other but 1st node  */
         if ( de->prv != NULL )
            error("mulExpr","de must be null or point to 1st node in expression!");

         /* now clean all terms but 1st out of this de expr */

         thisCur = de->nxt;
         while (thisCur != NULL)
          {
            nxt = thisCur->nxt;
            delExprTerm(&thisCur);
            thisCur = nxt;
          }

          /* then let de expr start replace thisNew start */
          de->prv = thisNew->prv;
          de->nxt = thisNew->nxt;
          updateTerm(&(de->tm),thisNew->tm.n,thisNew->tm.ps);
          delExprTerm(&thisNew);
          thisNew = NULL;
       }

   /* now sort all product expressions in each term */

   for ( thisCur = de; thisCur != NULL; thisCur = thisCur->nxt )
       sortps(thisCur->tm.ps);


    return 0;
}



int reduceExpr(expr_s **ex)
{
    expr_s *thisCur = NULL;
    expr_s *thisFor = NULL;
    expr_s *prv = NULL;
    expr_s *nxt = NULL;
    expr_s *nxtFor = NULL;
    int n_sum = 0;
```

```
    thisCur = *ex;
    if ( thisCur == NULL )
        return 0; /* nothing to do */

    /* back up to start of expression list */
    while (thisCur->prv != NULL )
      {
         error("reduceExpr","for now, shouldn't enter here");
         prv = thisCur->prv;
         thisCur = prv;
      }

     /* get the current term and check remainder of list for duplicates */
     while (thisCur->nxt != NULL)
      {
        thisFor = thisCur->nxt;
        /* first collect all matches, and accumulate result */
        n_sum = 0;
        while (thisFor != NULL )
          {
              if ( thisFor->tm.ps == NULL )
                  error("reduceExpr","thisFor tm.ps is NULL");
              if ( thisCur->tm.ps == NULL )
                  error("reduceExpr","thisCur tm.ps is NULL");

              if ( strcmp(thisFor->tm.ps, thisCur->tm.ps) == 0 )
                    n_sum += thisFor->tm.n;
              nxtFor = thisFor->nxt;
              thisFor = nxtFor;
          }
        /* next update the current term */
        thisCur->tm.n += n_sum;

        /* finally, delete all the forward references of the term */
        thisFor = thisCur->nxt;
        while (thisFor != NULL )
          {
              nxtFor = thisFor->nxt;
              if ( strcmp(thisFor->tm.ps, thisCur->tm.ps) == 0 )
                  delExprTerm(&thisFor);
              thisFor = nxtFor;
          }

        if ( (nxt = thisCur->nxt) == NULL )
            break;
        thisCur = nxt;
      }

    return 0;
}


int purgeExpr(expr_s **ex)
{
  /* remove all 0 terms from expr */
  expr_s *thisCur;
  expr_s *thisNew = NULL;
  expr_s *thisFirst;
  expr_s *prv;
  expr_s *nxt;
  expr_s *prvNew = NULL;
  expr_s *nxtNew = NULL;

  /* make a new list, attach it to 1st term node */

   if ( ex == NULL )
     error("purgeExpr","ex is null");

   thisFirst = *ex;
   if ( thisFirst == NULL )
     return 0;

   if ( thisFirst->nxt == NULL )
     {
       /* special case, only 1 term */
       if ( thisFirst->tm.n == 0 )
         updateTerm(&(thisFirst->tm),0,"#");
       return 0;
     }


   thisCur = thisFirst->nxt;
```

```
   while (thisCur != NULL)
    {
      nxt = thisCur->nxt;
       /* add back all terms to new list except ones with 0s */
      if ( thisCur->tm.n != 0 )
        {

         if ( (thisNew = newExprTerm(prvNew, (expr_s *) NULL)) == NULL )
              error("purgeExpr","unable to alloc newExprTerm");
         updateTerm(&(thisNew->tm),thisCur->tm.n,thisCur->tm.ps);
         if (prvNew != NULL)
              prvNew->nxt = thisNew;
         prvNew = thisNew;

        }
       thisCur = nxt;
    }

   if (thisNew == NULL)
    {
        /* put in a null node */
         if ( (thisNew = newExprTerm(prvNew, (expr_s *) NULL)) == NULL )
              error("purgeExpr","unable to alloc newExprTerm");
         updateTerm(&(thisNew->tm),0,"#");
         if (prvNew != NULL)
              prvNew->nxt = thisNew;
         prvNew = thisNew;

    }

   while (thisNew->prv != NULL)
    {
        prvNew = thisNew->prv;
        thisNew = prvNew;
    }

   if ( thisFirst->tm.n == 0 )
      updateTerm(&(thisFirst->tm),0,"#");

   thisCur = thisFirst->nxt;
   thisCur->prv = NULL;
   thisFirst->nxt = thisNew;
   thisNew->prv = thisFirst;
   while (thisCur != NULL)
    {
       nxt = thisCur->nxt;
       prv = thisCur->prv;
       delExprTerm(&thisCur);
       thisCur = nxt;
    }

   thisCur = thisFirst->nxt;
   if ( (thisFirst->tm.n == 0) && (thisCur != NULL) )
     {
       updateTerm(&(thisFirst->tm),thisCur->tm.n,thisCur->tm.ps);
       thisFirst->nxt = thisCur->nxt;
       if (thisCur->nxt != NULL) thisCur->nxt->prv = thisFirst;
       if (thisCur->tm.ps != NULL ) { free(thisCur->tm.ps); thisCur->tm.ps = NULL; }
       delExprTerm(&thisCur);
       thisCur = NULL;
     }


   thisCur = thisFirst->nxt;
   if ( (thisFirst->tm.n != 0) && (thisCur != NULL) )
     {
        if ( thisCur->tm.n == 0 )
         {
             thisFirst->nxt = thisCur->nxt;
             if ( thisCur->nxt != NULL ) thisCur->nxt->prv = thisFirst;
             if ( thisCur->tm.ps != NULL ) { free(thisCur->tm.ps); thisCur->tm.ps = NULL; }
             delExprTerm(&thisCur);
         }

     }

   return 0;
}


int sortExpr(expr_s *ex)
{
    int n; char *ps;
```

```
    expr_s *cur, *top;

    for ( top = ex; top != NULL; top = top->nxt )
     {
      for ( cur = top->nxt; cur != NULL; cur = cur->nxt )
        if ( top->tm.ps[0] != '#' ) /* keep '#' at front of expr */
          if ( (cur->tm.ps[0] == '#') || (strcmp(top->tm.ps,cur->tm.ps) > 0) )
            { /* swap terms -- just exchange n vals and pointers to str */
              n  = cur->tm.n;  cur->tm.n  = top->tm.n;  top->tm.n  = n;
              ps = cur->tm.ps; cur->tm.ps = top->tm.ps; top->tm.ps = ps;
            }
     }

    return 0;
}



int cleanExpr(int n, ... )
{
    /* remove all child nodes leaving top node intact */
    int i;
    expr_s *ex, *nxt, *theNxt;
    va_list argp;

    va_start(argp, n);
    for ( i = 0; i < n; i++ )
      {
         ex = va_arg(argp, expr_s *);
         if ( ex == NULL )
             error("cleanExpr","ex is null");
         nxt = ex->nxt; ex->nxt = NULL;
         while ( nxt != NULL )  {  theNxt = nxt->nxt;  delExprTerm(&nxt);  nxt = theNxt; }

         /* re-init the top node of expr */
         updateTerm(&(ex->tm),0,"#");
      }
     va_end(argp);

    return 0;
}
```

```
int initQuat(term_s *t, term_s *x, term_s *y, term_s *z, int n, ...)
{
   int i,j;   term_s * tms[4];   expr_s ** hexpr[4];   quat_s * q;
   va_list argp;

   tms[0] = t; tms[1] = x; tms[2] = y; tms[3] = z;

   va_start(argp, n);
   for ( i = 0; i < n; i++ )
   {
   q = va_arg(argp, quat_s *);

   hexpr[0] = &(q->t); hexpr[1] = &(q->x); hexpr[2] = &(q->y); hexpr[3] = &(q->z);

   for ( j = 0; j < 4; j++ )
     {
      if ( (*(hexpr[j]) = newExprTerm((expr_s *) NULL, (expr_s *) NULL)) == NULL )
         error("initQuat","newExprTerm failed");

      (*(hexpr[j]))->tm.n = tms[j]->n;

      if ( tms[j]->ps == NULL )
          (*(hexpr[j]))->tm.ps = NULL;
      else if ( ((*(hexpr[j]))->tm.ps = (char *) calloc(strlen(t->ps)+1,sizeof(char))) == NULL )
         error("initQuat","calloc failed on.tm.ps");
      else strcpy((*(hexpr[j]))->tm.ps,tms[j]->ps);
     }

   }
   va_end(argp);

    return 0;

}


int copyQuat(quat_s *dq, quat_s *sq)
{
    if ( dq == NULL || dq->t == NULL || dq->x == NULL || dq->y == NULL || dq->z == NULL )
      error("copyQuat","must call initQuat for destination dq before use in copyQuat.");

    /* now clean out expr in case of re-using old quat variable */
    cleanExpr(4, dq->t, dq->x, dq->y, dq->z);

    /* init or re-init dq */
    updateTerm(&(dq->t->tm),0,"#");
    updateTerm(&(dq->x->tm),0,"#");
    updateTerm(&(dq->y->tm),0,"#");
    updateTerm(&(dq->z->tm),0,"#");

    addQuat(dq,dq,sq);   /* copy by adding source "sq" to "dq=0" put result back in "dq" */

    return 0;

}

int addQuat(quat_s *qd, quat_s *q1, quat_s *q2)
{
    /* dq = q1 + q2 */
    addExpr(qd->t,q1->t,q2->t);
    addExpr(qd->x,q1->x,q2->x);
    addExpr(qd->y,q1->y,q2->y);
    addExpr(qd->z,q1->z,q2->z);
    reduceQuat(qd);
    purgeQuat(qd);
    sortQuat(qd);

    return 0;
}

int subQuat(quat_s *qd, quat_s *q1, quat_s *q2)
{
    /* dq = q1 - q2 */
    subExpr(qd->t,q1->t,q2->t);
    subExpr(qd->x,q1->x,q2->x);
    subExpr(qd->y,q1->y,q2->y);
    subExpr(qd->z,q1->z,q2->z);
    reduceQuat(qd);
    purgeQuat(qd);
    sortQuat(qd);

    return 0;
}
```

```
int mulQuat(quat_s *qd, quat_s *q1, quat_s *q2)
{
    /* qd = q1.q2  -- quaternion muliplication */
    expr_s t1,t2,t3,t4;

    /* init temp intermediate variables */
    t1.tm.ps = t2.tm.ps = t3.tm.ps = t4.tm.ps = NULL;
    t1.prv = t2.prv = t3.prv = t4.prv = NULL;
    t1.nxt = t2.nxt = t3.nxt = t4.nxt = NULL;

    updateTerm(&(t1.tm),0,"#");
    updateTerm(&(t2.tm),0,"#");
    updateTerm(&(t3.tm),0,"#");
    updateTerm(&(t4.tm),0,"#");

     /* t: */
     mulExpr(&t1,q1->t,q2->t);
     mulExpr(&t2,q1->x,q2->x);
     mulExpr(&t3,q1->y,q2->y);
     mulExpr(&t4,q1->z,q2->z);
     smulExpr(&t2,-1);
     smulExpr(&t3,-1);
     smulExpr(&t4,-1);
     addExpr(qd->t,&t1,&t2);
     addExpr(qd->t,qd->t,&t3);
     addExpr(qd->t,qd->t,&t4);

     /* now clean out temp expr lists : t1, t2, t3, t4 -- i.e. free mem */
     cleanExpr(4,&t1,&t2,&t3,&t4);

     /* x: */
     mulExpr(&t1,q1->t,q2->x);
     mulExpr(&t2,q1->x,q2->t);
     mulExpr(&t3,q1->y,q2->z);
     mulExpr(&t4,q1->z,q2->y);
     smulExpr(&t4,-1);
     addExpr(qd->x,&t1,&t2);
     addExpr(qd->x,qd->x,&t3);
     addExpr(qd->x,qd->x,&t4);

     /* now clean out temp expr lists : t1, t2, t3, t4 -- i.e. free mem */
     cleanExpr(4,&t1,&t2,&t3,&t4);

     /* y: */
     mulExpr(&t1,q1->t,q2->y);
     mulExpr(&t2,q1->y,q2->t);
     mulExpr(&t3,q1->z,q2->x);
     mulExpr(&t4,q1->x,q2->z);
     smulExpr(&t4,-1);
     addExpr(qd->y,&t1,&t2);
     addExpr(qd->y,qd->y,&t3);
     addExpr(qd->y,qd->y,&t4);

     /* now clean out temp expr lists : t1, t2, t3, t4 -- i.e. free mem */
     cleanExpr(4,&t1,&t2,&t3,&t4);

     /* z: */
     mulExpr(&t1,q1->t,q2->z);
     mulExpr(&t2,q1->z,q2->t);
     mulExpr(&t3,q1->x,q2->y);
     mulExpr(&t4,q1->y,q2->x);
     smulExpr(&t4,-1);
     addExpr(qd->z,&t1,&t2);
     addExpr(qd->z,qd->z,&t3);
     addExpr(qd->z,qd->z,&t4);

     /* now clean out temp expr lists : t1, t2, t3, t4 -- i.e. free mem */
     cleanExpr(4,&t1,&t2,&t3,&t4);

     /* we're done with temp variables, free remaining mem used */
     if (t1.tm.ps != NULL) { free(t1.tm.ps); t1.tm.ps = NULL; }
     if (t2.tm.ps != NULL) { free(t1.tm.ps); t1.tm.ps = NULL; }
     if (t3.tm.ps != NULL) { free(t1.tm.ps); t1.tm.ps = NULL; }
     if (t4.tm.ps != NULL) { free(t1.tm.ps); t1.tm.ps = NULL; }

     reduceQuat(qd);
     purgeQuat(qd);
     sortQuat(qd);

     return 0;
}
```

```
int smulQuat(quat_s *q, int s)      /* dq = s.q = q.s .. scalar mul -- the "s" is scalar */
{
  smulExpr(q->t,s); smulExpr(q->x,s); smulExpr(q->y,s); smulExpr(q->z,s);
  purgeQuat(q);  /* in case s=0, we remove all zeros after s.q */
  return 0;
}


int conjQuat(quat_s *q)      /* q <== q* -- conjugate q */
{
  smulExpr(q->x,-1); smulExpr(q->y,-1); smulExpr(q->z,-1);
  return 0;
}


int reduceQuat(quat_s *q)
{
  reduceExpr(&(q->t)); reduceExpr(&(q->x)); reduceExpr(&(q->y)); reduceExpr(&(q->z));
  return 0;
}


int purgeQuat(quat_s *q)
{
  purgeExpr(&(q->t)); purgeExpr(&(q->x)); purgeExpr(&(q->y)); purgeExpr(&(q->z));
  return 0;
}


int sortQuat(quat_s *q)
{
  sortExpr(q->t); sortExpr(q->x); sortExpr(q->y); sortExpr(q->z);
  return 0;
}


int printQuat(quat_s *q)
{
   printf("t: ");   printExpr(q->t);   printf("\n");
   printf("x: ");   printExpr(q->x);   printf("\n");
   printf("y: ");   printExpr(q->y);   printf("\n");
   printf("z: ");   printExpr(q->z);   printf("\n");
   printf("\n\n");
   return 0;
}



// .end
```

APPENDIX F: MATLAB code listings

The MATLAB code, scattered throughout the text, are all gathered here in one place for easy reference.

#-01-file: psolqabc.m

```
function psolqabc(A,B,C)
  global a b c;
% e.g:  A = [  2,  1,  0, -1 ];    %  a =  2 + 1i + 0j - 1k
% e.g:  B = [ -3,  0,  1,  1 ];    %  b = -3 + 0i + 1j + 1k
% e.g:  C = [  1,  0, -1,  1 ];    %  c =  1 + 0i - 1j + 1k

  a = [(A(1)+1i*A(2)),(A(3)+1i*A(4));-(A(3)+1i*A(4))',(A(1)+1i*A(2))'];
  b = [(B(1)+1i*B(2)),(B(3)+1i*B(4));-(B(3)+1i*B(4))',(B(1)+1i*B(2))'];
  c = [(C(1)+1i*C(2)),(C(3)+1i*C(4));-(C(3)+1i*C(4))',(C(1)+1i*C(2))'];
end
```

#-02-file: psolq.m

```
function up=psolq(r)
  global a b c;
  p = psolqd(r);
  pp = trace(p*p')/2;
  up = pp - 1;
end
```

#-03-file: psolqd.m; storage-file: psolqd_ver_1_0.m

```
function p=psolqd(r)      % ver: 1.0,  (aqq + bq + c = 0)
  global a b c;
  np = r^5*(r^4*a'*a*a' - c'*a*c')*(c*b' - r^2*b*a')*a;
  np = np + r*(c'*c*c' - r^4*a'*c*a')*(c*b' - r^2*b*a')*c;
  dp = trace((r^4*a*a'- c*c')*(r^2*a - c)*(r^2*a - c)'*(r^2*a + c)*(r^2*a + c)')/2;
  p = np/dp;
end
```

#-04-file: psolqv.m

```
function P=psolqv(r)
  global a b c;
  p = psolqd(r);
  P = [ real(p(1,1)) , imag(p(1,1)) , real(p(1,2)), imag(p(1,2)) ];
end
```

#-05-file: psolqck.m

```
function V=psolqck(r)
  global a b c;
  Q = r*psolqv(r);
  V = psolqev(Q);
end
```

#-06-file: psolqev.m; storage-file: psolqev_ver_1_0.m

```
function V=psolqev(Q)     % ver: 1.0,   (aqq + bq + c = 0)
  global a b c;
  q = [(Q(1)+1i*Q(2)),(Q(3)+1i*Q(4));-(Q(3)+1i*Q(4))',(Q(1)+1i*Q(2))'];
  v = a*q*q + b*q + c;
  V = [real(v(1,1)) , imag(v(1,1)), real(v(1,2)), imag(v(1,2))];
end
```

#-07-file: psolqcap.m

```
function up=psolqcap(r)
  global a b c;
  p = psolqd(r);
  pp = trace(p*p')/2;
  up = pp - 1; if (up > 1) up = 1;
end
```

#-08-file: psolqvh.m; storage-file: psolqvh_ver_1_0.m

```
function P=psolqvh(r)    % ver: 1.0,   (aqq + bq + c = 0)
 global a b c;
 ac = a'*c;
 acImagTest = abs(imag(ac(1,1))) + abs(real(ac(1,2))) + abs(imag(ac(1,2)));
 if ~(acImagTest == 0 && real(ac(1,1)) > 0)
   % use p = P1
   np = 5*r^4*(r^4*a'*a*a' - c'*a*c')*(c*b' - r^2*b*a')*a;
   np = np + r^5*(4*r^3*a'*a*a')*(c*b' - r^2*b*a')*a;
   np = np + r^5*(r^4*a'*a*a' - c'*a*c')*(-2*r*b*a')*a;
   np = np + (c'*c*c' - r^4*a'*c*a')*(c*b' - r^2*b*a')*c;
   np = np + r*(-4*r^3*a'*c*a')*(c*b' - r^2*b*a')*c;
   np = np + r*(c'*c*c' - r^4*a'*c*a')*(- 2*r*b*a')*c;
   dp = trace((4*r^3*a*a')*(r^4*a*a' - r^2*(c*a' + a*c') + c*c')*(r^4*a*a' + r^2*(c*a' + a*c') + c*c'))/2;
   dp = dp +  trace((r^4*a*a' - c*c')*(4*r^3*a*a' - 2*r*(c*a' + a*c'))*(r^4*a*a' + r^2*(c*a' + a*c') + c*c'))/2;
   dp = dp + trace((r^4*a*a' - c*c')*(r^4*a*a' - r^2*(c*a' + a*c') + c*c')*(4*r^3*a*a' + 2*r*(c*a' + a*c')))/2;
 else    % use p = P3
    np = -990*r^8*(a'*a*a'*b*a'*a) + 504*r^6*(a'*a*a'*c*b'*a) + 210*r^4*(c'*a*c'*b*a'*a + a'*c*a'*b*a'*c);
    np = np - 60*r^2*(c'*a*c'*c*b'*a + a'*c*a'*c*b'*c) - 6*(c'*c*c'*b*a'*c);
    dp = trace(96*r^3*(a*a')*(3*r^2*a*a' - (c*a' + a*c'))*(r^2*a + c)*(r^2*a' + c'))/2;
    dp = dp + trace(96*r^5*(a*a')*(2*r^2*a*a' - (c*a' + a*c'))*(2*r^2*a*a' + (c*a' + a*c')))/2;
    dp = dp + trace((r^2*a - c)*(r^2*a' - c')*(24*r*(a*a'))*(13*r^4*a*a' + 5*r^2*(c*a' + a*c') + c*c'))/2;
    r4a2c2 = real(trace(r^4*a*a' - c*c')/2);
    if r4a2c2 ~= 0   % add (r^4.|a|^2 - |c|^2)...part!
       dp1 = trace(48*r*(a*a')*(r^4*a*a' + c*c'))/2;
       dp1 = dp1 + trace(12*r*(2*r^2*a*a' - (c*a' + a*c'))*(6*r^2*a*a' + (c*a' + a*c')))/2;
       dp1 = dp1 + trace(12*r*(6*r^2*a*a' - (c*a' + a*c'))*(2*r^2*a*a' + (c*a' + a*c')))/2;
       dp = dp + r4a2c2*dp1;
    end
 end
 p = np/dp;
 P = [ real(p(1,1)), imag(p(1,1)), real(p(1,2)), imag(p(1,2)) ];
end
```

#-09-file: psolqvh0.m; storage-file: psolqvh0_ver_1_0.m

```
function P=psolqvh0      % ver: 1.0,  (aqq + bq + c = 0)
 global a b c;
 ac = a'*c;
 acImagTest = abs(imag(ac(1,1))) + abs(real(ac(1,2))) + abs(imag(ac(1,2)));
 r = sqrt(sqrt(trace(c*c')/trace(a*a')));
 I = [1,0;0,1];
 mac = sqrt(real(trace(ac*ac')/2)); % = |a*.c|
 if acImagTest == 0 && real(ac(1,1)) > 0    % a*.c in [R], a*.c > 0
    % use P = P3
    p = (b'*a - 2*a'*b)/2;
 else
    % use P = P1
    p = ((5*a'*c - 3*mac*I)*b'*a + (a*a')*(I - 3*a'*c/mac)*b'*c)/(2*mac - 2*real(ac(1,1)));
    p = ( p - 5*a'*b )/4;
 end
 p = (1/r)*p/real(trace(a*a')/2);
 P = [ real(p(1,1)), imag(p(1,1)), real(p(1,2)), imag(p(1,2)) ];
end
```

#-10-file: psolqvh0.m; storage-file: psolqvh0_ver_1_1.m

```
function P=psolqvh0      % ver: 1.1,  (aqq + bq + c = 0)
 global a b c;
 cc = c; bb = b; % save global vars
 I = [1,0;0,1]; c = (a^-1)*c; b = (a^-1)*b; % rescale
 cImagTest = abs(imag(c(1,1))) + abs(real(c(1,2))) + abs(imag(c(1,2)));
```

```
 mc = sqrt(real(trace(c*c')/2)); % = |c|
 r = sqrt(mc); % = rt|c|
 if  cImagTest == 0 && real(c(1,1)) > 0  % a*.c in [R], a*.c > 0
    % use P = P3
    p = b'/2 - b;
 else
    % use P = P1
    p = (((5*c - 3*mc*I)*b' + (I - 3*c/mc)*b'*c)/(2*mc - 2*real(c(1,1))) - 5*b)/4;
 end
p = (1/r)*p;
P = [ real(p(1,1)) , imag(p(1,1)), real(p(1,2)), imag(p(1,2)) ];
c = cc; b = bb; % restore global vars
end
```

#-11-file: psolqd.m; storage-file: psolqd_ver_1_1.m

```
function p=psolqd(r)    % ver: 1.1,  (aqq + bq + c = 0)
 global a b c;
 p = (r*(r^2*a*a' - c')*c*b'*(r^2*a' - c')^-1 - r^3*a'*b)/(trace(r^4*a*a' - c*c')/2);
end
```

#-12-file: psolqd.m; storage-file: psolqd_ver_1_2.m

```
function p=psolqd(r)    % ver: 1.2,  (aqq + bq + c = 0)
 global a b c;
 cc = c; bb = b; % save global vars
 I = [1,0;0,1];
 c = (a^-1)*c; b = (a^-1)*b;
 p = (r*(r^2*I - c')*c*b'*(r^2*I - c')^-1 - r^3*b)/(trace(r^4*I - c*c')/2);
 c = cc; b = bb; % restore global vars
end
```

#-13-file: pvec4.m; storage-file: pvec4_ver_1_0.m

```
function v=pvec4     % ver: 1.0,   (aqq + bq + c = 0)
  global a b c;
  cc = c;  bb = b; % save global vars
  c = (a^-1)*c;  b = (a^-1)*b; % rescale
  % compute poly coefs
  a3 = -b*b';
  a2 = -2*c*c' + (b*c'*b + b'*c*b');
  a1 = -(b*b')*(c*c');
  a0 = (c*c')*(c*c');
  % extract scalar vals from 2x2 matrices
  s3 = real(a3(1,1));
  s2 = real(a2(1,1));
  s1 = real(a1(1,1));
  s0 = real(a0(1,1));
  c = cc;  b = bb;  % restore global vars
  v = [1,s3,s2,s1,s0];
end
```

#-14-file: pvec6.m; storage-file: pvec6_ver_1_0.m

```
function v=pvec6     % ver: 1.0,   (aqq + bq + c = 0)
  global a b c;
  cc = c;  bb = b; % save global vars
  c = (a^-1)*c;  b = (a^-1)*b; % rescale
  % compute poly coefs
  a5 = -b*b' - (c + c');
  a4 = -c*c' + (b*b')*(c + c') + (b*c'*b + b'*c*b');
  a3 = -2*(c*c')*(b*b' - (c + c')) - (b*c'*b + b'*c*b')*(c + c') - (b*c - c*b)^2;
  a2 = (-(c*c') + (b*b')*(c + c') + (b*c'*b + b'*c*b'))*(c*c');
  a1 = (-(b*b') - (c + c'))*(c*c')*(c*c');
  a0 = (c*c')*(c*c')*(c*c');
  % extract scalar vals from 2x2 matrices
  s5 = real(a5(1,1));
  s4 = real(a4(1,1));
  s3 = real(a3(1,1));
```

```
  s2 = real(a2(1,1));
  s1 = real(a1(1,1));
  s0 = real(a0(1,1));
  c = cc;  b = bb;  % restore global vars
  v = [1,s5,s4,s3,s2,s1,s0];
end
```

#-15-file: psolqvc.m; storage-file: psolqvc_ver_1_0.m

```
function P=psolqvc(A,B,C,r)
% Bi-Quaternion Unit Direction Function ver: 1.0, (aqq + bq + c = 0)
%
%
%     r.(r^2.a* - c*).(cb*.(r^2.a - c) - r^2.(r^2.a - c).a*b)
% p = ------------------------------------------------------------
%               (r^4.|a|^2 - |c|^2).|r^2.a - c|^2
%
%
 NP1 = r*(r^2*conq(A) - conq(C));
 NP2 = mulq(C,mulq(conq(B),(r^2*A - C))) - r^2*mulq((r^2*A - C),mulq(conq(A),B));
 NP  = mulq(NP1,NP2);
 DP1 = (r^4*mulq(A,conq(A)) - mulq(C,conq(C)));
 DP2 = mulq(r^2*A - C, conq(r^2*A - C));
 DP  = mulq(DP1,DP2);
 P   = NP/DP(1);
end
```

#-16-file: aqqbqc.m; storage-file: aqqbqc_ver_1_0.m

```
function [Q1 Q2]=aqqbqc(A,B,C)
% Quaternion Quadratic Equation Solver ver: 1.0, (aqq + bq + c = 0)
  global a b c
  psolqabc(A,B,C)
  QInf = [Inf, Inf, Inf, Inf];

  if a == 0      % linear eqn.
     % bq + c = 0
     q = -b^(-1)*c;
     Q1 = [ real(q(1,1)), imag(q(1,1)), real(q(1,2)), imag(q(1,2)) ];
     Q2 = QInf;  % recall: q = [w,v;-v',w'], w = t + x*1i, v = y + z*1i
     return;     %         q = t + xi + yj + zk
  end
  if c == 0      % linear eqn, + 0
     % (aq + b)q = 0
     q = -a^(-1)*b;
     Q1 = [ 0, 0, 0, 0]; % by convention |Q1| <= |Q2| !
     Q2 = [ real(q(1,1)), imag(q(1,1)), real(q(1,2)), imag(q(1,2)) ];
     return;
  end
  r = sqrt(sqrt(trace(c*c')/trace(a*a')));
  ac = a'*c;
  acImagTest = abs(imag(ac(1,1))) + abs(real(ac(1,2))) + abs(imag(ac(1,2)));
  %  = abs(x) + abs(y) + abs(z) % ; t + xi + yj + zk

  if b == 0     % one-step methods
    % aq^2 + c = 0
    if acImagTest == 0   % i.e. a*.c in [R]
       if real(ac(1,1)) > 0 % i.e. a*.c > 0
         % Case 1: "out of scope " infinite # of roots
         % pick an arbitary pair of valid pure quaternion roots
         % q = r*[0+1i,1+1i;-1+1i,0-1i]/sqrt(3);
         % =or= pick a 'random' opposite pair of valid pure quaternion roots
         n = -1 + 2*rand([1,3]); n = n/sqrt(n*n');
         q = r*[0+n(1)*1i,n(2)+n(3)*1i;-n(2)+n(3)*1i,0-n(1)*1i];
       else     %  i.e. a*.c < 0
         % Case 2: real roots
         q = r*[1,0;0,1];
       end
    else   % a*.c is in [H]
      q = r*([1,0;0,1] - ac/sqrt(abs(trace(ac*ac'))/2));
      if real(ac(1,1)) == 0
         % Case 3: S(a*.c) = 0
         q = q/sqrt(2);
      else
         % Case 4: S(a*.c) != 0
         q = q/sqrt(2 - 2*real(ac(1,1))/sqrt(abs(trace(ac*ac'))/2));
```

```
      end
    end
  Q1 = [ real(q(1,1)), imag(q(1,1)), real(q(1,2)), imag(q(1,2)) ];
  Q2 = -Q1;
  return;
  end % .of b == 0 cases.

  % b != 0 cases : two-step methods
  p = psolqv(r-10^-5);
  pp = p*p';
  if pp > 1
     r1 = fzero(@psolq,[0,r-10^-5]);
     r2 = fzero(@psolq,[r+10^-5,10^23]);
     p1 = psolqv(r1);
     p2 = psolqv(r2);
  else
    % offplane magic factor solution method 'third step'
    P = psolqvh0;  % L'Hospital's rule evaluate P=p(r=sqrt(|c|/|a|))
    Q = r*P;       % avg. root
    E = mulq(A,mulq(Q,Q)) + mulq(B,Q) + C;  % quadratic residual
    cA = -A; cA(1) = A(1);  % conjugate of A
    W = r^2*mulq(cA,E);    % weighted residual quaternion
    wImagTest = abs(W(2)) + abs(W(3)) + abs(W(4));
    if wImagTest == 0   % W in [R]
      if W(1) == 0
        % Case 0:
        % soln found ! two roots both equal to avg root!
        Q1 = Q;
        Q2 = Q1;
        return;
      elseif W(1) > 0
        % Case 1: W in [R], W > 0.
        % infinite # of roots
        % pick arbitrary valid pure quaternion unit for u
        % U = [0,1,1,1]/sqrt(3);
        % =or= pick a 'random' valid pure quaternion unit
        n = -1 + 2*rand([1,3]); n = n/sqrt(n*n');
        U = [0,n(1),n(2),n(3)];
      else % W(1) < 0
        % Case 2: W in [R], W < 0.
        U = [1,0,0,0];
      end
   else % W in [H]
      U = ([1,0,0,0] - W/sqrt(W*W'));
      if W(1) == 0
         % Case 3: W in [H], S(W) = 0.
         U = U/sqrt(2);
      else
         % Case 4: W in [H], S(W) != 0.
         U = U/sqrt(2 - 2*W(1,1)/sqrt(W*W'));
      end
    end
  % calc the magic !
  fixup = sqrt(sqrt((E*E')/(C*C')));
  p1 = P - U*fixup;
  p2 = P + U*fixup;
  r1 = r;
  r2 = r;
  end
  Q1 = r1*p1;
  Q2 = r2*p2;
end
```

#-17-file: aqqbqc.m; storage-file: aqqbqc_ver_1_1.m

```
function [Q1 Q2]=aqqbqc(A,B,C)   % by pmj
% Quaternion Quadratic Equation Solver  ver: 1.1,  (aqq + bq + c = 0)
 QZro = [  0,  0,  0,  0];
 QInf = [Inf,Inf,Inf,Inf];
 QNaN = [NaN,NaN,NaN,NaN];
 vTOm = @(V) [V(1)+1i*V(2),V(3)+1i*V(4);-V(3)+1i*V(4),V(1)-1i*V(2)];
 mTOv = @(m) [real(m(1,1)), imag(m(1,1)), real(m(1,2)), imag(m(1,2)) ];
 a = vTOm(A); b = vTOm(B); c = vTOm(C);

 if      ((a == 0) & (b == 0) & (c == 0));  Q1 = QInf;  Q2 = QInf; return;
 elseif  ((a == 0) & (b == 0) & (c ~= 0));  Q1 = QNaN;  Q2 = QNaN; return;
 elseif  ((a == 0) & (b ~= 0));  Q1 = mTOv(-(b^-1)*c);  Q2 = QInf; return;
 elseif  ((a ~= 0) & (c == 0));  Q1 = QZro;  Q2 = mTOv(-(a^-1)*b); return;
 else  % ((a ~= 0) & (c ~= 0));  % ** by convention |Q1| <= |Q2| **
   I = [1,0;0,1]; b = (a^-1)*b; c = (a^-1)*c; % lead reduce params
```

```
  r = sqrt(sqrt(trace(c*c')/2));
 end

 pm = @(r) ((r*(r^2*I-c')*c*b'*(r^2*I-c')^-1) - r^3*b)/(trace(r^4*I-c*c')/2);
 pf = @(r) mTOv(pm(r))*mTOv(pm(r))' - 1;
 ImagTest = @(w) abs(imag(w(1,1))) + abs(real(w(1,2))) + abs(imag(w(1,2)));

 P = mTOv(pm(r-10^-5)); PP = P*P';

 if PP > 1
   r1 = fzero(pf,[0,r-10^-5]);
   r2 = r*r/r1;
   Q1 = r1 * mTOv(pm(r1));
   Q2 = r2 * mTOv(pm(r2));
   return
 end
 % PP <= 1
 if b == 0
    p = [0,0;0,0];
 else
    mc = sqrt(abs(trace(c*c')/2)); % mc = |c|
    if ImagTest(c) == 0 && real(c(1,1)) > 0     % use p = P3;
       p = (1/r)*( (1/2)*b' - b );
    else       % use p = P1;
       p = (1/r)*(1/4)*(((5*c - 3*mc*I)*b' + (I - 3*c/mc)*b'*c)/(2*mc - 2*real(c(1,1))) - 5*b);
   end
 end

 q = r*p; Q = mTOv(q); % avg. root
 w = q*q + b*q + c; W = mTOv(w);

 if ImagTest(w) ~= 0;   U = ([1,0,0,0] - W/sqrt(W*W'))/sqrt(2 - 2*W(1)/sqrt(W*W'));
 elseif W(1) < 0;       U = [1,0,0,0];
 elseif W(1) == 0;      U = [0,0,0,0]; % equal roots
 else % W(1) > 0;    infinite # roots, pick valid pair at random!
    n = -1 + 2*rand([1,3]); n = n/sqrt(n*n');  U = [0,n];
 end
 U = U*sqrt(sqrt(W*W')); % fixup
 Q1 = Q - U;
 Q2 = Q + U;
end % .of fn aqqbqc()
```

#-18-file: mulq.m

```
function V=mulq(A,B)
V = [ A(1)*B(1) - A(2)*B(2) - A(3)*B(3) - A(4)*B(4), ...
      A(1)*B(2) + A(2)*B(1) + A(3)*B(4) - A(4)*B(3), ...
      A(1)*B(3) + A(3)*B(1) + A(4)*B(2) - A(2)*B(4), ...
      A(1)*B(4) + A(4)*B(1) + A(2)*B(3) - A(3)*B(2)  ];
end
```

#-19-file: invq.m

```
function V=invq(A)
  AA = A(1)*A(1) + A(2)*A(2) + A(3)*A(3) + A(4)*A(4);
  V = [ A(1), -A(2), -A(3), -A(4) ]/AA;
end
```

#-20-file: conq.m

```
function V=conq(A)
  V = [ A(1), -A(2), -A(3), -A(4) ];
end
```

APPENDIX G: Summary of Unit Quaternion Solutions

$$a, b, c, q \in \mathbb{H}\_R$$

$$aq^2 + bq + c = 0 \ , \qquad b \neq 0, \ |q| = 1$$
$$q = \frac{(|a|^2 a^* - c^* a c^*)(cb^* a - ba^* a) + (|c|^2 c^* - a^* c a^*)(cb^* c - ba^* c)}{(|a|^2 - |c|^2)((|a|^2 + |c|^2)^2 - ((ac^*) + (ac^*)^*)^2)} \qquad \text{(g-1)}$$

$$q^2 a + qb + c = 0 \ , \qquad b \neq 0, \ |q| = 1$$
$$q = \frac{(|a|^2 a - c a^* c)(b^* c a^* - a^* b a^*) + (|c|^2 c - a c^* a)(b^* c c^* - a^* b c^*)}{(|a|^2 - |c|^2)((|a|^2 + |c|^2)^2 - (ac^* + (ac^*)^*)^2)} \qquad \text{(g-2)}$$

$$qaq + bq + c = 0 \ , \qquad b \neq 0, \ |q| = 1$$
$$q \ = \ \frac{(cab^* - aa^* b)(|a|^2 a^* - cac) + (c^* cb^* - a^* c^* b)(|c|^2 c - a^* c^* a^*)}{(|a|^2 - |c|^2) \cdot ((|a|^2 + |c|^2)^2 - ((ca) + (ca)^*)^2)} \qquad \text{(g-3)}$$

$$qaq + qb + c = 0 \ , \qquad b \neq 0, \ |q| = 1$$
$$q \ = \qquad \text{(g-4)}$$

(working paper 01 – to be completed)

[H1] Hamilton, William R., *On The Connexion of Quaternions with Continued Fractions and Quadratic Equations*, Proceedings of the Royal Irish Academy, 5 (1853), pp. 219-221, 299-301.

[YT1] Tian, Yongge *The equations $ax - xb = c$, $ax - \bar{x}b = c$, and $\bar{x}ax = b$ in quaternions.* Southeast Asian Bulletin of Mathematics 28, 343-362., 2004

[PJ2] Jack, Peter M. *Hexpentaquaternions: a two-hand quaternion algebra*, Jan 29, 2006.

[PJ3] Jack, Peter M. *Quatro-Quaternions and the matrix representations of octonions*, Jul 02, 2006.

[PJ4] Jack, Peter M. *General solutions to linear problems in quaternion variables.*, Nov 29, 2007.